2003

SRL2003 IJCAI 2003 Workshop on Learning Statistical Models from Relational Data

Lise Getoor
University of Maryland

SRL2003

IJCAI 2003 Workshop on Learning Statistical Models from Relational Data

Organizers

Lise Getoor, University of Maryland, College Park
David Jensen, University of Massachusetts, Amherst

Program Committee

James Cussens, University of York, UK
Luc De Raedt, Albert-Ludwigs-University, Germany
Pedro Domingos, University of Washington, USA
Kristian Kersting, Albert-Ludwigs-University, Germany
Stephen Muggleton, Imperial College, London, UK
Avi Pfeffer, Harvard University, USA
Taisuke Sato, Tokyo Institute of Technology, Japan
Lyle Ungar, University of Pennsylvania, USA

Preface

This workshop is the second in a series of workshops held in conjunction with AAAI and IJCAI. The first workshop was held in July, 2000 at AAAI. Notes from that workshop are available at http://robotics.stanford.edu/srl/. Since the AAAI 2000 workshop, there has been a surge of interest in this area. The efforts have been diffused across a wide collection of sub-areas in computer science including machine learning, database management and theoretical computer science. Much of the work is organized around applications such as hypertext mining, modeling the World Wide Web and social network analysis. This surge in interest has been fueled by the large interest in the Internet and web mining and interest in mining social networks for counter-terrorism and intelligence (led by DARPA's Evidence Extraction and Link Discovery program).

We are excited to see common topics and themes emerging from the various research camps and cliques. We hope to use this year's workshop to begin formulating a general theory for statistical relational learning (and perhaps devise a better term for it, too!).
This year's workshop will consist of the following sessions:

- Feature Construction, Aggregation, and Propositionalization
- Link Prediction
- Identity Uncertainty, Record Linkage and Consolidation
- Instances vs. Classes
- General Relations/Handling Time and Space
- Models and Systems

The sessions will be devoted largely to discussion; there will be no formal paper presentations.

A key to the success of the workshop will be the use of a collaboratively edited website to encourage participation before the workshop, to support interaction during the workshop and to provide a record after the workshop. The website is:

http://kdl.cs.umass.edu/srl2003/

The login and password are: srl2003/ijcai2003!

We'd like to give extend our sincere thanks to the program committee, the authors, and all of the participants.

We are looking forward to a lively and productive workshop in Acapulco,

Lise Getoor, University of Maryland, College Park
David Jensen, University of Massachusetts, Amherst

Table of Contents

Research Papers

Research Summaries and Statements of Interest

Supporting Material

Speeding Up Multi-Relational Data Mining

Anna Atramentov, Vasant Honavar
Artificial Intelligence Research Laboratory,
Computer Science Department, Iowa State University,
226 Atanasoff Hall, Ames, IA 50011-1040, USA,
{anjuta, honavar}@cs.iastate.edu

Abstract

We present a general approach to speeding up a family of multi-relational data mining algorithms that construct and use *selection graphs* to obtain the information needed for building predictive models (e.g., decision tree classifiers) from relational database. Preliminary results of our experiments suggest that the proposed method can yield 1-2 orders of magnitude reductions in the running time of such algorithms without any deterioration in the quality of results. The proposed modifications enhance the applicability of multi-relational data mining algorithms to significantly larger relational databases that would otherwise be not feasible in practice.

1 Introduction

Recent advances in high throughput data acquisition, digital storage, and communications technologies have made it possible to gather very large amounts of data in many scientific and commercial domains. Much of this data resides in relational databases. Even when the data repository is not a relational database, it is often convenient to view heterogeneous data sources as if they were a collection of relations [Reinoso-Castillo, 2002] for the purpose of extracting and organizing information from multiple sources. Thus, the task of learning from relational data has begun to receive significant attention in the literature [Blockeel, 1998; Knobbe *et al.*, 1999a; Friedman *et al.*, 1999; Koller, 1999; Krogel and Wrobel, 2001; Getoor, 2001; Kersting and De Raedt, 2000; Pfeffer, 2000; Dzeroski and Lavrac, 2001; Dehaspe and Raedt, 1997; Jaeger, 1997; Karalic and Bratko, 1997].

Recently, [Knobbe *et al.*, 1999a] outlined a general framework for multi-relational data mining which exploits structured query language (SQL) to gather the information needed for constructing classifiers (e.g., decision trees) from multi-relational data. Based on this framework, several algorithms for multi-relational data mining have been developed. Experiments reported by [Leiva, 2002] have shown that MRDTL – a multi-relational decision tree learning algorithm is competitive with other approaches to learning from relational data. One common feature of all algorithms based on the multi-relational data mining framework proposed by [Knobbe *et al.*, 1999a] is their use of *selection graphs* to query the relevant databases to obtain the information (e.g., statistics) needed for constructing a model. Our experiments with MRDTL revealed that the execution of queries encoded by such selection graphs was a major bottleneck in terms of the running time of the algorithm. Hence, this paper describes an approach for significantly speeding up some of the most time consuming components of such algorithms. Preliminary results of our experiments suggest that the proposed method can yield one to two orders of magnitude speedups in the case of MRDTL. We expect similar speedups to be obtained with other multi-relational data mining algorithms which construct and use selection graphs.

The rest of the paper is organized as follows: in Section 2 we overview multi-relational data-mining framework, in Section 3 we describe the speed up scheme for this framework and in Section 4 we show the experimental results that we obtained applying the scheme.

2 Multi-Relational Data Mining

2.1 Relational Databases

A relational database consists of a set of tables $D = \{X_1, X_2, ...X_n\}$, and a set of associations between pairs of tables. In each table a row represents description of one record. A column represents values of some attribute for the records in the table. An attribute A from table X is denoted by $X.A$.

Definition 2.1 *The* domain *of the attribute $X.A$ is denoted as $DOM(X.A)$ and is defined as the set of all different values that the records from table X have in the column of attribute A.*

Associations between tables are defined through primary and foreign key attributes in D.

Definition 2.2 *A* primary key *attribute of table X, denoted as $X.ID$, has a unique value for each row in this table.*

Definition 2.3 *A* foreign key *attribute in table Y referencing table X, denoted as $Y.X\_ID$, takes values from $DOM(X.ID)$.*

An example of a relational database is shown in Figure 1. There are three tables and three associations between tables. The primary keys of the tables GENE, COMPOSITION, and INTERACTION are: GENE_ID, C_ID, and

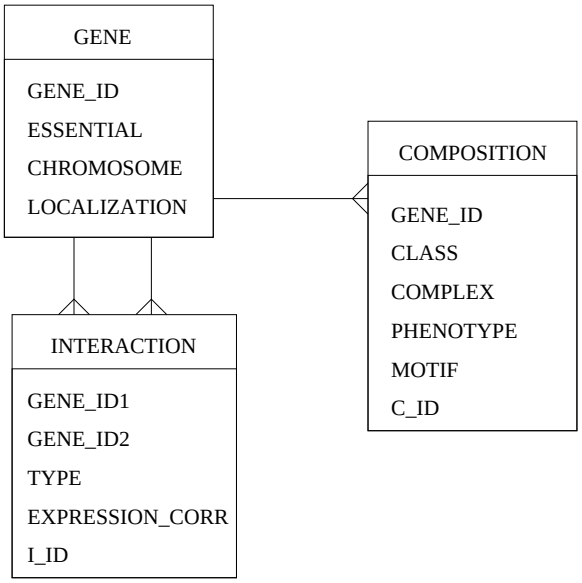


Figure 1: Example database

I_ID, respectively. Each COMPOSITION record references some GENE record through the foreign key COMPOSITION.GENE_ID, and each INTERACTION record references two GENE records through the foreign keys INTERACTION.GENE_ID1 and INTERACTION.GENE_ID2.

In this setting, if an attribute of interest is chosen, it is called *target attribute*, and the table in which this attribute is stored is called *target table* and is denoted by T_0.

Each record in T_0 corresponds to a single object. Additional information about an object is stored in other tables of the database, which can be looked up, when following the associations between tables.

2.2 Multi-Relational Data Mining Framework

Multi-relational data mining framework is based on the search for interesting patterns in the relational database, where multi-relational patterns can be viewed as "pieces of substructure encountered in the structure of the objects of interest" [Knobbe *et al.*, 1999a].

Definition 2.4 *A multi-relational object is* covered *by a multi-relational pattern iff the substructure described by the multi-relational pattern, in terms of both attribute-value conditions and structural conditions, occurs at least once in the multi-relational object. ([Knobbe* et al.*, 1999a])*

Multi-relational patterns also can be viewed as subsets of the objects from the database having some property. The most interesting subsets are chosen according to some measure (i.e. information gain for classification task), which guides the search in the space of all patterns. The search for interesting patterns usually proceeds by a top-down induction. For each interesting pattern, subputterns are obtained with the help of refinement operator, which can be seen as further division of the set of objects covered by initial pattern. Top-down induction of interesting pattern proceeds recursively applying such refinement operators to the best patterns.

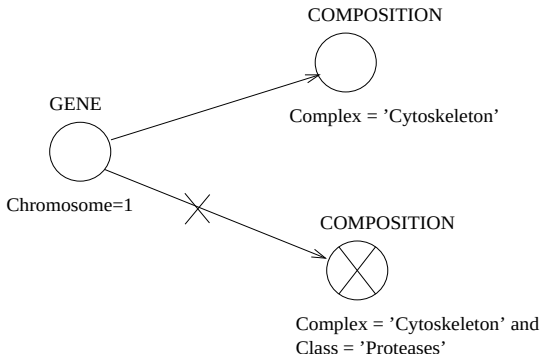


Figure 2: Selection graph, corresponding to those GENE(s) that belong to chromosome number 1, that have at least one COMPOSITION record whose complex value is 'Cytoskeleton', but for which none of the COMPOSITION records have complex value 'Cytoskeleton' and class value 'Proteases' at the same time.

Multi-relational pattern language is defined in terms of *selection graphs* and *refinements* which are described in the following sections.

2.3 Selection Graphs

Multi-relational patterns are expressed in a graphical language of selection graphs [Knobbe *et al.*, 1999b].

Definition 2.5 *A* selection graph S *is a directed graph* $S = (N, E)$. N *represents the set of nodes in* S *in the form of tuples* (X, C, s, f)*, where* X *is a table from* D*,* C *is the set of conditions on attributes in* X *(for example,* X*.color = 'red' or* X*.salary* $> 5{,}000$*),* s *is a flag with possible values* open *and* closed*, and* f *is a flag with possible values* front *and* back*.* E *represents edges in* S *in the form of tuples* (p, q, a, e)*, where* p *and* q *are nodes and* a *is a relation between* p *and* q *in the data model (for example, X.ID = Y.X_ID), and* e *is a flag with possible values* present *and* absent*. The selection graph should contain at least one node* n_0 *that corresponds to the target table* T_0*.*

An example of the selection graph for the data model from Figure 1 is shown in Figure 2. This selection graph corresponds to those GENE(s) that belong to chromosome number 1, that have at least one COMPOSITION record whose complex value is 'Cytoskeleton', but for which none of the COMPOSITION records have complex value 'Cytoskeleton' and class value 'Proteases' at the same time. In this example the target table is GENE, and within GENE the target attribute is LOCALIZATION.

In graphical representation of a selection graph, the value of s is represented by the presence or absence of a cross in the node, representing the value *open* and *closed*, respectively. The value for e, in turn, is indicated by the presence (*present* value) or absence (*absent* value) of a cross on the corresponding arrow representing the edge. An edge between nodes p and q chooses the records in the database that match the joint condition, a, between the tables which is defined by the relation between the primary key in p and a foreign key

in q, or the other way around. For example, the join condition, a, between table GENE and COMPOSITION in selection graph from Figure 2 is GENE.GENE_ID = COMPOSITION.GENE_ID.

A *present* edge between tables p and q combined with a list of conditions, $q.C$ and $p.C$, selects those objects that match the list of conditions, $q.C$ and $p.C$, and belong to the join between p and q, specified by join condition, $e.a$. On the other hand, an *absent* edge between tables p and q combined with a list of conditions, $q.C$ and $p.C$, selects those objects that match condition $p.C$ but do not satisfy the following: match $q.C$ and belong to the join between tables at the same time.

Flag f is set to *front* for those nodes that on their path to n_0 have no closed edges. For all the other nodes flag f is set to *back*.

[Knobbe *et al.*, 1999b] introduces the algorithm (Figure 3) for translating a selection graph into SQL query that returns the records in the target table covered by this selection graph, where subgraph(S, $j.q$) procedure returns the subgraph of the selection graph S starting with the node q as the target node, with label s reset to *open*, removing the part of the graph that was connected to this node with the edge j and reseting all the values of flag f at the resulting selection graph by definition of f. Notation $j.q$.key means the name of the attribute (primary or foreign key) in the table q that is associated with the table p in relation $j.a$.

```
TRANSLATE(S, key)
Input Selection graph S, key (primary or foreign) in the
target node of S
Output SQL query for creating sufficient information about
graph S
 1   table_list := ''
 2   condition_list := ''
 3   join_list := ''
 4   for each node i in S do
 5       if (i.s = 'open' and i.f = 'front')
 6           table_list.add(i.table_name + 'T' + i)
 7           for each condition c in i do
 8               condition_list.add(c)
 9   for each edge j in S do
10       if (j.e = 'present')
11           if (j.q.s = 'open' and j.q.f = 'front')
12               join_list.add(j.a)
13       else
14           join_list.add(
                 j.p + '.' + j.p.primary_key + ' not in ' +
                 TRANSLATE( subgraph(S, j.q), j.q.key))
15   return 'select distinct' + 'T0.' + key +
         ' from ' + table_list +
         ' where ' + join_list + ' and ' + condition_list
```

Figure 3: Translation of selection graph into SQL query

Using this procedure the graph in Figure 2 translates to the SQL statement shown in Figure 4.

```
select   distinct T0.gene_id
from     GENE T0, COMPOSITION T1
where    T0.gene_id = T1.gene_id
         and T0.chromosome = 1
         and T1.complex = 'Cytoskeleton'
         and T0.gene_id not in
         ( select T0.gene_id
         from COMPOSITION T0
         where T0.complex = 'Cytoskeleton'
             and T0.class = Proteases)
```

Figure 4: SQL query corresponding to the selection graph in Figure 2

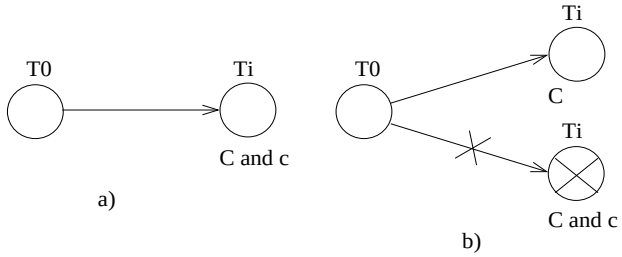


Figure 5: Complement refinements for adding condition to the node: a) positive condition, b) negative condition

2.4 Refinements of the Selection Graphs

Multi-relational data mining algorithms search for and successively refine interesting patterns and select promising ones based on some impurity measure (e.g. information gain). The set of refinements introduced by [Knobbe *et al.*, 1999b] are given below. Note that all of these refinements can only be applied to the *open*, *front* nodes in the selection graph S.

- Add positive condition (Figure 5 a)). This refinement will simply add a condition c to the set of conditions C in the node T_i of selection graph S without actually changing the structure of S.
- Add negative condition (Figure 5 b). If the node which is refined is not n_0, this refinement will introduce a new absent edge from the parent of the selection node in question. The condition list of the selection node will be copied to the new closed node, and will be extended by the new condition. This node will also get the copies of the children of the selection graph in question and open edges to those children will be added. If the node which is refined does represent the target table, the condition is simply negated and added to the current list of conditions for this node. This refinement is complement to the "add positive condition refinement", in the sense that it covers those objects from the original selection graph which were not covered by corresponding "add positive condition" refinement.
- Add present edge and open node (Figure 6 a)). This refinement introduces a present edge together with its corresponding table to the selection graph S.

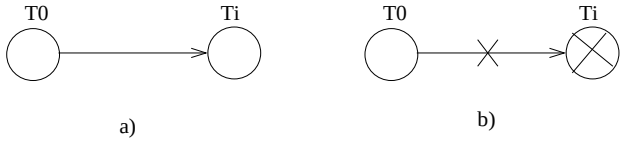


Figure 6: Complement refinements for adding edge to selection graph: a) adding present edge and open node, b) adding absent edge and closed node

- Add absent edge and closed node (Figure 6 b). This refinement introduces an absent edge together with its corresponding table to the selection graph S. This refinement is complement to the "add present edge and open node", in the sense that it covers those objects from the original selection graph which were not covered by "add present edge and open node" refinement.

It is important to note that only through the "add edge" refinements the exploration of all the tables in the database is done. We can consider "add condition" refinement on some attribute from some table only after the edge to that table has been added to the selection graph. What happens if the values of the attributes in some table are important for the task but the edge to this table can never be added, i.e. adding edge doesn't result in further split of the data covered by the refined selection graph? Look ahead refinements, which are a sequence of several refinements, are used for dealing with this situation. In the case when some refinement doesn't split the data covered by the selection graph, the next set of refinements is also considered as refinements of the original selection graph.

3 Speeding Up Multi-Relational Data Mining

Let S be some selection graph. Any refinement of S covers the subset of instances covered by S. Since all the refinements of S usually need to be examined, storing intermediate results obtained from S will reduce the amount of time needed to examine all its refinements.

The goal of this section is to show what intermediate information should be stored for each selection graph S so that the instances covered by each of its refinements can be recovered quickly.

The knowledge of the structure of the selection graph S is enough to restore all the objects in the database corresponding to any refinement R of S. This can be done by first applying the refinement to S to obtain a refined selection graph $R(S)$, which in turn is then transformed into an SQL query as described in Subsection 2.3. The size of the resulting SQL query increases with the complexity of the graph, resulting in the corresponding increase in the execution time of the query.

It is possible to substantially speed up this step of the algorithm as follows. For each object covered by selection graph S we store only its class label and the primary key values from the tables corresponding to the *open*, *front* nodes in the selection graph S. We call the resulting table the sufficient table for S and denote it by I_S.

The procedure that transforms selection graph S into SQL query for creating sufficient table I_S is shown in Figure 7.

```
SUF_TABLE(S)
Input Selection graph S
Output SQL query for creating sufficient table I_S
 1   table_list, condition_list, join_list :=
         extract_from(TRANSLATE(S))
 2   primary_key_list := 'T_0.target_attribute'
 3   for each node i in S do
 4       if (i.s = 'open' and i.f = 'front')
 5           primary_key_list .add(i.ID)
 6   return 'create table I_S as ' +
         ' (select ' + primary_key_list +
         ' from ' + table_list +
         ' where ' + join_list +
         ' and ' + condition_list + ')'
```

Figure 7: Algorithm for generating SQL query corresponding to the sufficient table I_S of the selection graph S

Given a sufficient table I_S, we can restore all the records from the target table that are covered by the selection graph S, by applying the following SQL query on table I_S:
select distinct T_0.primary_key **from** I_S.

The sufficient table I_S stores all the records corresponding to the selection graph S, i.e., all the records satisfying the constraints imposed by S, even though these constrains are not explicit anymore.

Let R be a refinement of the selection graph S, and $R(S)$ a new selection graph resulting from refining S with R. The procedure for obtaining the sufficient table for $R(S)$ given I_S is shown in Figure 8.

The sufficient table for a selection graph contains all the information necessary to obtain the database objects that are covered by the selection graph and any of its refinements.

Proposition 1 *Given a selection graph S, its sufficient table I_S, and a refinement R, the table constructed by REFINEMENT_SUF_TABLE(I_S, R) will contain the same records as the table constructed by SUF_TABLE($R(S)$)*

Proof sketch: Selection graph can be viewed as multi-relational pattern consisting of two subpatterns. The one that corresponds to all the *open*, *front* nodes in the selection graph, and the complement one. Let's denote the first subpattern as EXPLICIT subpattern, and the latter as IMPLICIT subpattern. The sufficient table contains the information about the EXPLICIT subpattern. Information about IMPLICIT subpattern is hidden in the sufficient table. It is important to note though, that objects stored in sufficient table still match IMPLICIT subpattern. Refinements can be applied only to the *open*, *front* nodes. Let's consider applying either 'add positive condition' refinement or 'add present edge' refinement. The resulting refined selection graph consists of extended (refined) EXPLICIT subpattern and unchanged IMPLICIT subpattern. This means that applying the refinement only to the sufficient table (as it is done in Figure 8) will result in objects matching to the extended EXPLICIT subpattern and inherently matching to the IMPLICIT subpattern, which means that they are matching to the refined selection graph. Similar

```
REFINEMENT_SUF_TABLE(I_S, R)
Input Sufficient table I_S for selection graph S,
refinement R
Output SQL query for sufficient table for R(S)
 1   table_list := 'I'_S
 2   condition_list := ''
 3   join_list := ''
 4   primary_key_list := primary_keys(I_S)
 5   if R == add positive condition, c, in table T_i
 6       table_list += 'T'_i
 7       condition_list += 'T_i.c'
 8       join_list += T_i.ID+' ='+I_S.T_i_ID
 9   else if R == add negative condition, c, in table T_i
10       condition_list += T_0.ID + 'is not in
             ( select distinct' + I_S.T_0_ID +
             ' from' + I_S, T_i +
             ' where' + T_i.c + 'and' + T_i.ID +
                 ' =' +I_S.T_i_ID+')'
11   else if R = add present edge, e, from T_i to T_j
12       table_list += T_i+',' +T_j
13       join_list += T_i.ID+' =' +I_S.T_i_ID+
             ' and ' + e.a
14       primary_key_list += T_j.ID
15   else if R == add closed edge, e from T_i to T_j
16       condition_list += T_0.ID + 'is not in
         ( select distinct' + I_S.T_0_ID +
         ' from ' + I_S+',' +T_i+',' +T_j +
         ' where ' + T_i.ID+' =' +I_S.T_i_ID+
             ' and ' + e.a+')'
17   return 'create table I_R as ' +
         '(select ' + primary_key_list +
         ' from ' + table_list +
         ' where ' + join_list +
         ' and ' + condition_list + ')'
```

Figure 8: Algorithm for generating SQL query corresponding to sufficient table $I_{R(S)}$

argument can be used for the case of other refinements. ∎

Note that REFINEMENT_SUF_TABLE procedure always returns a query of the constant size, i.e. the number of tables that need to be joint and the number of conditions that need to be applied is constant, which means that the time needed for executing this query doesn't increase with the size of the selection graph. On the other hand, the time needed for the execution of the TRANSLATE(S) function increases considerably with the size of the selection graph.

The above discussion can be extended to the look-ahead refinements, since they are a sequence of two refinements.

4 Experimental Results

We illustrate how the proposed approach can speed up a multi-relational data mining algorithm by considering multi-relational decision tree learning (MRDTL) algorithm, which constructs a decision tree for classifying a target attribute from a target table in a given database.

This algorithm proposed in [Knobbe *et al.*, 1999b] and implemented in [Leiva, 2002] is an extension of the logical decision tree induction algorithm called TILDE proposed by [Blockeel, 1998]. Essentially, MRDTL, like the propositional version of the decision tree algorithm [Quinlan, 1993], adds decision nodes to the tree through a process of successive refinement until some termination criterion is met (e.g., correct classification of instances in the training set). The choice of the decision node to be added at each step is guided by a suitable impurity measure (e.g., information gain). MRDTL starts with the selection graph containing a single node at the root of the tree, which represents the set of all objects of interest in the relational database. This node corresponds to the target table T_0. The algorithm iteratively considers every possible refinement that can be made to the current pattern (selection graph) S with respect to the database D and selects, in a greedy fashion, the optimal refinement (i.e., the one that maximizes information gain) and its complement.

Each candidate refinement is evaluated in terms of the split of the data induced by it with respect to the target attribute, as in the case of the propositional version of the decision tree learning algorithm [Quinlan, 1993]. Splits based on numerical attributes are handled using a technique similar to that of C4.5 algorithm [Quinlan, 1993] with modifications proposed in [Fayyad and Irani, 1992; Quinlan, 1996].

The hypothesis resulting from the induction of the relational decision tree algorithm described above can be viewed as a set of SQL queries associated with the selection graphs that correspond to the leaves of the decision tree. Each selection graph (query) has a class label associated with it. If the corresponding node is not a pure node, (i.e., it misclassifies some of the training instances that match the query), the label associated with the node can be based on the classification of the majority of training instances that match the corresponding selection graph. Alternatively, we can use probabilistic assignment of labels based on the distribution of class labels among the training instances that match the corresponding selection graph. The complementary nature of the different branches of a decision tree ensures that a given instance will not be assigned conflicting labels. It is also worth noting that it is not necessary to traverse the entire tree in order to classify a new instance; all the constraints on a certain path are stored in the selection graph associated with the corresponding leaf node. Instances that do not match the selection graphs associated with any of the leaf nodes in the tree are assigned unknown label and are counted as incorrectly classified when evaluating the accuracy of the tree on test data.

We have implemented MRDTL in Java using Oracle relational database and tested it on different databases. We have also implemented the speedup scheme for this algorithm. The resulting algorithm is shown in Figure 9.

We conducted our experiments on the data for prediction gene localization from KDD Cup 2001 [Cheng *et al.*, 2002]. Our current implementation of MRDTL assumes that the target table has a primary key, therefore it was necessary to normalize one of the initial tables given in this task. This normalization was achieved by creating tables named GENE, INTERACTION, and COMPOSITION as shown in Figure 1. For the gene/protein localization task, the target table is

Tree_Induction(D, S, I_S)
Input Database D, selection graph S, sufficient table I_S
Output The root of the tree, T

```
1   ALL := all_refinements(S)
2   R := optimal_refinement(I_S, D, ALL)
3   if stopping_criteria(I_S)
4       return leaf
5   else
6       T_left := Tree_Induction(D, R(S), R(I_S))
7       T_right := Tree_Induction(D, R̄(S), R̄(I_S))
8       return node(T_left, T_right, R)
```

Figure 9: MRDTL algorithm with speed up

	o_r_min	o_r_max	o_r_all	all
WOSU	0.04	70.642	3838.512	4764.15
WSU	0.00	3.656	65.241	416.74

Table 1: Experimental results. Here o_r_min denotes the shortest running times (in seconds) spent by the algorithm on a single call of *optimal_refinement* procedure, o_r_max denotes the longest running times (in seconds) spent by the algorithm on a single call of *optimal_refinement* procedure, o_r_all denotes the running time (in seconds) spent by the algorithm on all calls of the *optimal_refinement* procedure, all denotes the overall running time (in seconds) of the algorithm, WOSU denotes the results for the run of the algorithm without speed up scheme implemented, and WSU denotes the results for the run of the algorithm with speed up scheme implemented.

GENE and the target attribute is LOCALIZATION. The resulting training set consists of 862 genes and the test set consists of 381 genes. We constructed a classifier using all the training data and test the resulting classifier on the test set.

We have recorded the running times of the algorithm with and without speedup scheme proposed in the paper. We also measured the amount of time spent on the function *optimal_refinement*.

Experimental results are shown in Table 1, where o_r_min denotes the shortest running times (in seconds) spent by the algorithm on a single call of *optimal_refinement* procedure, o_r_max denotes the longest running times (in seconds) spent by the algorithm on a single call of *optimal_refinement* procedure, o_r_all denotes the running time (in seconds) spent by the algorithm on all calls of the *optimal_refinement* procedure, all denotes the overall running time (in seconds) of the algorithm, WOSU denotes the results for the run of the algorithm without speed up scheme implemented, and WSU denotes the results for the run of the algorithm with speed up scheme implemented.

The overall running time spent on querying the database in training phase was decreased by a factor of around 59. The running time improvement by a factor of 11 was observed in the overall running time for the MRDTL algorithm on this database. Some calls of *optimal_refinement* procedure had running time improvement up to a factor of 1000.

5 Conclusion

In this paper we present a general approach to speeding up a class of multi-relational data mining algorithms. We have incorporated the proposed method into MRDTL algorithm. Preliminary results of our experiments have shown that the proposed method yields one to two orders of magnitude reductions in the running time of the algorithm. The proposed modifications make it feasible to apply multi-relational data mining algorithms to significantly larger relational databases. Our work in progress is aimed at:

- Incorporation of sophisticated methods for handling missing attribute values into MRDTL
- Incorporation of sophisticated pruning methods or complexity regularization techniques into MRDTL to minimize overfitting and improve generalization
- More extensive experimental evaluation of MRDTL on real-world data sets
- Development of ontology-guided multi-relational decision tree learning algorithms to generate classifiers at multiple levels of abstraction (based on the recently developed prepositional decision tree counterparts of such algorithms [Zhang *et al.*, 2002]
- Development of variants of MRDTL for classification tasks where the classes are not disjoint, based on the recently developed propositional decision tree counterparts of such algorithms [Caragea *et al.*, in preparation]
- Development of variants of MRDTL that can learn from heterogeneous, distributed, autonomous data sources based on recently developed techniques for distributed learning [Caragea *et al.*, 2001b; 2001a] and ontology-based data integration [Honavar et al., 2001; Honavar et al., 2002; Reinoso-Castillo, 2002].
- Application of multi-relational data mining algorithms to data-driven knowledge discovery problems in bioinformatics and computational biology.

6 Acknowledgements

This research was supported in part by a grant from the National Science Foundation (NSF ITR 021969) and a research assistantship funded by the Iowa State University Graduate College. The paper has benefited from discussions with Hector Leiva and Doina Caragea of the Iowa State University Artificial Intelligence Research Laboratory.

References

[Blockeel, 1998] Hendrik Blockeel. *Top-down induction of first order logical decision trees*. PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, 1998.

[Caragea *et al.*, 2001a] D. Caragea, A. Silvescu, and V. Honavar. Decision tree learning from distributed data. Technical Report TR, Iowa State University, Ames, IA, 2001.

[Caragea *et al.*, 2001b] D. Caragea, A. Silvescu, and V. Honavar. *Invited Chapter. Toward a Theoretical Framework for Analysis and Synthesis of Agents That Learn from Distributed Dynamic Data Sources*. Berlin: Springer-Verlag, 2001.

[Caragea *et al.*, in preparation] D. Caragea, A. Silvescu, and V. Honavar. Learning decision tree classifiers when the classes are not disjoint, in preparation.

[Cheng *et al.*, 2002] J. Cheng, M. Krogel, J. Sese, Hatzis C., S. Morishita, H. Hayashi, and D. Page. Kdd cup 2001 report. In *ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations*, volume 3, 2002.

[Dehaspe and Raedt, 1997] L. Dehaspe and L. De Raedt. Mining association rules in multiple relations. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297, pages 125–132. Springer-Verlag, 1997.

[Dzeroski and Lavrac, 2001] S. Dzeroski and N Lavrac. Relational data mining. Springer-Verlag, 2001.

[Fayyad and Irani, 1992] U. M. Fayyad and K. B Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8, 1992.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, , and Pfeffer. Learning probabilistic relational models. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*. Morgan Kaufman, 1999.

[Getoor, 2001] L Getoor. Multi-relational data mining using probabilistic relational models: research summary. In *Proceedings of the First Workshop in Multi-relational Data Mining*, 2001.

[Jaeger, 1997] M Jaeger. Relational bayesian networks. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-1997)*, 1997.

[Karalic and Bratko, 1997] Karalic and Bratko. First order regression. *Machine Learning 26*, 1997.

[Kersting and De Raedt, 2000] K. Kersting and L. De Raedt. Bayesian logic programs. In *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, 2000.

[Knobbe *et al.*, 1999a] J. Knobbe, H. Blockeel, A. Siebes, and Van der Wallen D. Multi-relational data mining. In *Proceedings of Benelearn 99*, 1999.

[Knobbe *et al.*, 1999b] J. Knobbe, H. Blockeel, A. Siebes, and Van der Wallen D. Multi-relational decision tree induction. In *Proceedings of the 3rdEuropean Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 99*, 1999.

[Koller, 1999] D Koller. Probabilistic relational models. In S. Dzeroski and P. Flach, editors, *Proceedings of 9th International Workshop on Inductive Logic Programming (ILP-99)*. Springer, 1999.

[Krogel and Wrobel, 2001] M. Krogel and S. Wrobel. Transformation-based learning using multirelational aggregation. In Celine Rouveirol and Michele Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming*, volume 2157 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[Leiva, 2002] Hector Ariel Leiva. A multi-relational decision tree learning algorithm. *M.S. thesis. Deparment of Computer Science. Iowa State University*, 2002.

[Pfeffer, 2000] A. Pfeffer. A bayesian language for cumulative learning. In *Proceedings of AAAI 2000 Workshop on Learning Statististical Models from Relational Data*. AAAI Press, 2000.

[Quinlan, 1993] R. Quinlan. *C4.5: Programs for Machine Learning*. 1993.

[Quinlan, 1996] R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4, 1996.

[Reinoso-Castillo, 2002] Jaime Reinoso-Castillo. Ontology-driven information extraction and integration fromheterogeneous distributed autonomous data sources. *M.S. Thesis. Department of Computer Science. Iowa State University*, 2002.

[Zhang *et al.*, 2002] J. Zhang, A. Silvescu, , and V. Honavar. Ontology-driven induction of decision trees at multiple levels of abstraction. In *Proceedings of the Symposium on Abstraction, Reformulation, and Approximation (SARA-2002)*, Kananaskis, Alberta, Canada,, 2002.

The Relational Vector-space Model and Industry Classification

Abraham Bernstein
University of Zurich
Winterthurerstrasse 190
8057 Zurich, Switzerland
bernstein@ifi.unizh.ch

Scott Clearwater
Clearwater Ways
P.O. Box 620513
Woodside, CA 94062, U. S. A.
clearway@ix.netcom.com

Foster Provost
NYU Stern School of Business
44 West 4th Street, Room 8-86
New York, NY 10012, U.S.A.
fprovost@stern.nyu.edu

ABSTRACT

This paper addresses the classification of linked entities. We introduce a relational vector-space (VS) model (in analogy to the VS model used in information retrieval) that abstracts the linked structure, representing entities by vectors of weights. Given labeled data as background knowledge/training data, classification procedures can be defined for this model, including a straightforward, "direct" model using weighted adjacency vectors. Using a large set of tasks from the domain of company affiliation identification, we demonstrate that such classification procedures can be effective. We then examine the method in more detail, showing that as expected the classification performance correlates with the relational autocorrelation of the data set. We then turn the tables and use the relational VS scores as a way to analyze/visualize the relational autocorrelation present in a complex linked structure. The main contribution of the paper is to introduce the relational VS model as a potentially useful addition to the toolkit for relational data mining. It could provide useful constructed features for domains with low to moderate relational autocorrelation; it may be effective by itself for domains with high levels of relational autocorrelation, and it provides a useful abstraction for analyzing the properties of linked data.

Keywords

relational data mining, vector-space models, industry classification, homophily, relational autocorrelation, relational-neighbor classifier

1. INTRODUCTION

The analysis of linked data differs from the traditional data-mining scenario: the data items, instead of being statistically independent, have relationships to each other. Linked data are ubiquitous, and relational data mining is receiving increasing attention with the explicit linking of web sites, and with the need to analyze social networks for applications such as counterterrorism [1, 2, 3]. We address a particular relational data mining application: identifying the group membership of linked entities. We address company-industry affiliation, but the framework and methods we describe are intended to be general.

Figure 1 shows a link diagram of companies and their relationships, as extracted from the business news. Colors indicate industry-sector affiliation. The diagram suggests that relationships may play a useful role in identifying the (unknown) affiliation of a company, because linked companies often have the same affiliation.

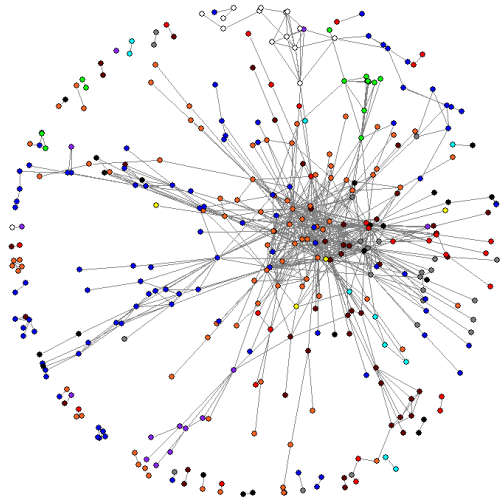


Figure 1. Link diagram of firms. Only links with strength > 4 are shown (but proximity also indicates relatedness). Colors indicate industry-sector membership.

The key contribution of this paper is the presentation and demonstration of a simple, but useful, method for producing classification models from linked data. In analogy to information retrieval [4], we represent entities using a vector-space model. The *relational vector-space* (RVS) model abstracts away much of the graph structure, representing entities by adjacency vectors. Various classification procedures can be defined on the RVS model.

The main attraction of the RVS model is its simplicity. We argue that RVS class-membership scores could be useful constructed features for more complex (relational) data-mining approaches, such as ILP [5] that do not naturally summarize the class membership of local neighborhoods. We also believe that for certain tasks, the RVS model may be appropriate by itself.

The rest of the paper is organized as follows. We present the RVS model formally, and use it to define several classification scoring functions. Next we introduce the domain of company affiliation identification, from which we will take a set of classification tasks. Then we present the results of an experimental case study, examining the effectiveness of the RVS model for classification in this domain. Finally, we show how the model's scores can be used to analyze and visualize certain class-related information about the original, complex graph.

2. THE RVS MODEL

We make a direct analogy to the "vector-space model" used for information retrieval, in which all textual and linguistic structure is ignored and documents are represented by vectors of weights on words. The relational vector-space model is a similarly limited abstraction of the graph structure, into a representation on which straightforward classification techniques can be built. Specifically, each dimension in the vector space corresponds to another entity; each entity is represented by a (weighted) adjacency vector (i.e., the magnitude along each dimension is some measure of the strength of the relationship).

2.1 General Model

Formally, we consider a set of entities E and a set $B \subseteq E$ of "background knowledge" entities. Later in our company affiliation domain, the entities will be companies and the background knowledge will be companies for which the classification is known. We place an (arbitrary) ordering on B, resulting in b_i, i = 1, .. |B|. These define the dimensions of the vector space, and thereby the dimensions along which any entity can be described.

Definition: An entity e is described by an *entity vector* $\mathbf{w} = (w_1, w_{2,} \ldots)$, where w_i is the strength of the relationship between entity e and background entity b_i. Ignoring strengths gives a *simple entity vector*, **w,** where the w_i are binary (presence/absence of a link).

This relational vector-space representation can be used for classification and clustering of entities, and other tasks that rely on entity similarity. In this paper, we will consider entity classification. In particular, consider a discrete, finite set of classes $\mathbf{C}$, such that for each $C_i \in \mathbf{C}$, $C_i \subseteq E$. If $e \in C_i$, e is considered to be a member of class i. In principle, the classes need not be mutually exclusive, but we will consider them to be for this paper, so the class can be considered to be a single-valued attribute of an entity and (later) we can adapt previous notions of relational autocorrelation directly. By definition, for $e \in B$, class membership is known. We would like to determine (estimate) class membership for at least one entity $e \notin B$.

Definition: Each class $C_i \in \mathbf{C}$ is described by a *class vector* $\mathbf{c}_i = (c_{i,1}, c_{i,2,} \ldots)$, where $c_{i,j}$ is the strength of the relationship between class C_i and background entity b_i.

In order to classify an entity, we will consider how similar the entity vector is to each class vector, using a similarity-based scoring function. First, let us define a generalized scoring function.

Definition: The *generalized RVS score* of entity e for class i is the normalized inner product of $\mathbf{w}$ and $\mathbf{c_i}$ (the normalizing function $g(\mathbf{w}, \mathbf{c}_i)$ is discussed below):

$$d(e,i) = \frac{\mathbf{w} \cdot \mathbf{c}_i}{g(\mathbf{w}, \mathbf{c}_i)}$$

RVS scores may be used for classification and other class-based scoring (e.g., for ranking) directly. They also could provide generally useful constructed features to be used by other methods (for example, more complex relational data mining methods [1,2,3]).

2.2 Instantiating the RVS Model

To define specific RVS scores we must answer three questions, which we now will address in turn.

1. How exactly are the entity vectors, $\mathbf{w}$, defined?
2. How exactly are the class vectors, $\mathbf{c_i}$, defined?
3. What normalizing function, $g(\mathbf{w}, \mathbf{c}_i)$ is used?

Entity vectors. Recall that an entity vector is composed of the strengths of the relationships between the entity e and the background entities b_i. Of course, the definition of strength is domain dependent, but there are some general issues worth highlighting. In all cases, we will consider $w_i = 0$ to indicate the lack of a relationship between e and b_i. A simple way of defining entity vectors is to ignore strengths, creating a vector of binary indicators. If there is a natural notion of strength, such as the number of links between entities, this gives an obvious way of defining the w_i. However, in analogy to how the vector-space model is used in text classification, a TFIDF-like weighting scheme [4] may be provide added discrimination power.

Class vectors. Defining class vectors is somewhat more involved. One general *direct* method is to give non-zero weights to the background entities that are members of the class. The distribution of weights places an *a priori* directionality on the class vector, which ideally maximizes discriminability. Using uniform weights defines a set of simple, "canonical" vectors for each class.

Definition: The *canonical class vector*, $\mathbf{c_i}$, for class i has non-zero components:

$$c_{i,j} = 1 \Leftrightarrow b_j \in C_i$$

Other distributions of direct weights may be natural for a particular domain, based on background knowledge or statistics summarized from the corpus of background entities. For company affiliation classification, companies in an industry (class) may be weighted by

market capitalization or by a measure of marginal probability of linkage to same-class companies.

These direct methods assume that linkage to members of the same class is sufficient for discrimination. It may be that members of the same class are not linked to each other, but are linked to the same other entities (or other classes). Short of abandoning the RVS approach for a more complex graph-based approach, an *indirect* method for defining class vectors may be beneficial.

Definition: The *simple indirect class vector*, $\mathbf{sic_i}$, for class i is the vector sum of the entity vectors for the background entities belonging to the class:

$$\mathbf{sic_i} = \sum_{e \in C_i \cap B} \mathbf{w}$$

One can define more complicated indirect class vectors. For example, a class centroid would be slightly more complicated. An even more complicated indirect method would be to redefine the b_i, one per class, as "super-entities." Then an indirect method could compare an entity's distribution of links to the various super-entities to the average distributions for those classes. For this paper, we do not consider complicated variations further.

Normalization functions. Generally, $g(\mathbf{w}, \mathbf{c}_i)$ defines the semantics of the similarity represented by the score. For example, the familiar "cosine similarity" between the entity vector and the class vector is d(e,i) with the following normalization function:

$$g(\mathbf{w}, \mathbf{c}_i) = \|\mathbf{w}\| \left\|\mathbf{c}_i\right\|,$$

where $\| \ \|$ is the Euclidean (L2) norm. Whether the exact cosine distance, or some other normalization, is appropriate is domain dependent, but also depends on the definitions of $\mathbf{w}$ and $\mathbf{c_i}$. For the experiments below, we will look at several scoring functions representing different similarities. These scoring functions are defined by different instantiations of $\mathbf{w}$, $\mathbf{c_i}$, and $g(\mathbf{w}, \mathbf{c}_i)$.

2.3 Five RVS scoring functions

The RVS model gives a convenient design space of classification scoring functions. We concentrate on the canonical class vector, because it is easy to define, and creates intuitively attractive scores (that perform well in our domain).

Definition: The *class-normalized direct RVS score* of entity e for class i is the inner product of $\hat{\mathbf{w}}$ and the canonical class vector $\mathbf{c_i}$, normalized by the L1 norm of $\mathbf{c}$.

$$s_{cnd}(e,i) = \frac{\hat{\mathbf{w}} \cdot \mathbf{c}_i}{\sum c_{i,j}}$$

The class-normalized direct RVS score counts up the connected entities belonging to the class, and then normalizes by the size of the class,[1] so that certain classes do not get higher scores simply because they are larger.

Definition: The *entity-normalized direct RVS score* of entity e for class i is the inner product of $\hat{\mathbf{w}}$ and the canonical class vector $\mathbf{c_i}$, normalized by the L1 norm of $\hat{\mathbf{w}}$.

$$s_{end}(e,i) = \frac{\hat{\mathbf{w}} \cdot \mathbf{c}_i}{\sum \hat{\mathrm{w}}_j}$$

The entity-normalized direct RVS score is attractive intuitively: it represents the proportion of connected entities that are members of C_i. This normalizes so that certain entities do not get higher scores simply by being more highly connected.

Definition: The *weighted, entity-normalized direct (wend) RVS score* of entity e for class i is the inner product of $\mathbf{w}$ and the canonical class vector $\mathbf{c_i}$, normalized by the L1 norm of $\mathbf{w}$.

$$s_{wend}(e,i) = \frac{\mathbf{w} \cdot \mathbf{c}_i}{\sum w_j}$$

Using a weighted entity vector inherently deals with noise (spurious, low-weight links) in the data. Using the L1 norm of the weight vector gives the intuitively appealing weighted proportion of links that are to members of the class of interest.

All three of these methods directly relate the entity vectors $\mathbf{w}$ with the respective canonical class vectors $\mathbf{c_i}$. A second group of scoring functions relates the entity vector $\mathbf{w}$ with the simple indirect class vector $\mathbf{sic_i}$ of a class.

Definition: The *(simple) indirect RVS score* of entity e for class i is the cosine similarity between $\mathbf{w}$ and $\mathbf{sic_i}$,

$$d(e,i) = \frac{\mathbf{w} \cdot \mathbf{sic}_i}{\|\mathbf{w}\| \|\mathbf{sic}_i\|}$$

We define *efigf* weights (entity frequency inverse graph frequency) analogously to the TFIDF (text frequency inverse document frequency) weights used in Information Retrieval [4].

Definition: The *efigf-based indirect RVS score* of entity e for class i is the cosine between the efigf-normalized vector $\mathbf{w}'$ and the analogously normalized vector $\mathbf{sic_i'}$, where

$$\mathbf{ef} = w \frac{1}{\max_l(w_l)}, \ igf_i = \log\left(\frac{N}{n_i}\right), \text{ and}$$

$$\mathbf{w'} = \mathbf{ef} \ ? \ \mathbf{igf} \ (\mathrm{sic_i'} \text{ analogously})$$

$$\text{hence, } d_{efigf}(e,i) = \frac{\mathbf{w}' \cdot \mathbf{sic}_i'}{\|\mathbf{w}'\| \|\mathbf{sic}_i'\|}$$

[1] For the canonical class vector, the semantics of the cosine of the angle between it and a weighted entity vector is dubious.

3. DOMAIN & TASKS

To demonstrate the RVS model, we report a case study involving several classification tasks from the domain of company affiliation identification. Identifying the group membership of companies is a prerequisite for solving various problems. Consider industry membership. Determining which companies belong to a particular industry is essential for intellectual property (e.g., patent) litigation, financial analysis (e.g., balancing a portfolio, constructing sector funds), making/improving government economic projections, and so on.

Traditionally, industry membership has been determined by a manual process, and there are various existing classifications. For example, the US Government's Office of Management and Budget has developed a framework for how to assign SIC codes ("Standard Industry Classification" codes–hierarchical, four digit codes used as industry identifiers for firms). Business information companies, such as Hoover's and Yahoo, have different industry classifications (which often do not have a high degree of correspondence with the assigned SIC codes). There are known problems with industry classifications. For example, one study showed that two common SIC-code sources for the same companies disagreed on more than 36% of the codes at the 2-digit code level, and on more then 80% at the 4-digit level [6].

The RVS model can take as background knowledge any industry classification, and (attempt to) classify companies based on it. This gives the additional flexibility to adjust the classification of some background companies, and have the model adjust the rest accordingly, or start from scratch with a new scheme.

The quality of the generalization performance is an empirical question, which we address next for Yahoo's classification. Thus, for the RVS model, E is the set of companies, **C** comprises the Yahoo classifications (industry sector, unless otherwise noted), and B contains the companies for which the Yahoo classification is (deemed to be) known. We chose Yahoo because the granularity of the classifications (12 sectors) was attractive for a conference-paper study and because of ease of access to the data.

For the RVS model we also need a source for links between companies. For this study we chose a generic, but easily accessible link: two companies are linked if they cooccur in a business news story, with the strength of the relationship being the number of such links. Note that cooccurrence lumps together a wide variety of relationships, including joint ventures, mergers/acquisitions, product-related, market related, and so on. Some have nothing to do with industry membership (e.g., two companies happen to announce earnings on the same day). We based the cooccurrences on a collection of news stories from the period December 1999 to September 2002, for which the news provider had assigned at least two ticker symbols and for which the symbols appeared in the Yahoo classification.

4. RESULTS

To compare the various RVS scoring methods, we take each affiliation (the 12 Yahoo sectors) and ask how well the companies can be separated into those belonging to the affiliation and those not. We examine the five scoring functions listed in Section 2.2. and two extensions (described later). We also examined the methods using as the affiliations 97 Yahoo industries, with similar results (which we also use for illustration).

4.1 ROC Analysis for Sectors

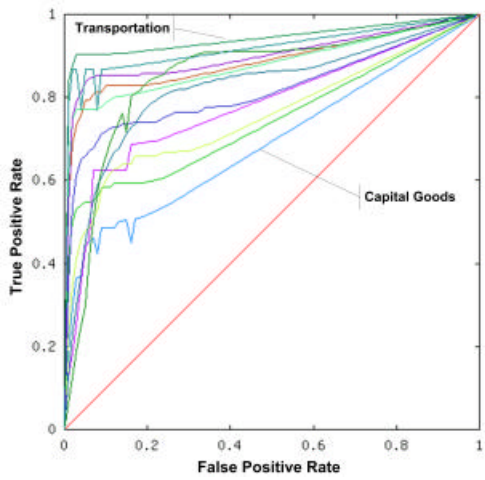


Figure 2: ROC curve for weighted, entity-normalized method (averaged over 10 runs)

We use ROC analysis [7, 8] to assess the model's ability to separate class members from non-members. For a given scoring of companies, ROC curves plot all the possible tradeoffs between correctly classifying the members of the class (the true positive rate, on the y-axis) and incorrectly identifying non-members of the class (false-positive rate, on the x-axis). The area under the ROC curve (AUC), equivalent to the Wilcoxon-Mann-Whitney statistic, is the probability that a member of the class will be scored higher than a non-member [9]. Error is calculated as 1 – AUC, and since the AUCs often are close to 1, relative error reduction[2] is reported for comparisons.

Figure 2 shows the ROC curves for the best method, the weighted, entity-normalized direct score (s_{wend}). Generalization performance ranges from moderate class separability (AUC=0.68 for Capital Goods) to excellent class separability (0.93 for Transportation). Referring back to Figure 1, Transportation is green, and we can see

[2] Relative error reduction of method2 over method1 = (AUC2 – AUC1)/(1-AUC1).

that green nodes are very well interlinked. (Capital Goods, cyan, are interlinked not nearly as well.)

Table 1 reports the AUCs of all 5 scoring functions for the 12 classification tasks. In most cases all the scoring methods classify considerably better than random (represented by the diagonal in ROC space). s_{wend} consistently performs better than the other scores (with only a few exceptions).. Table 2 shows the relative error reduction of s_{wend} over the other methods. s_{wend} has lower error than its closest competitor, the simple s_{end}, on 10 of 12 classification tasks, but achieves only a 2.3% error reduction on average.

	area under curve				
Sector	s_{end}	s_{cnd}	s_{wend}	d_{si}	d_{efigf}
BasicMaterials	0.7318	0.6644	0.7339	0.6218	0.6494
CapitalGoods	0.6781	0.6635	0.6810	0.5274	0.5476
Conglomerates	0.7563	0.5318	0.7697	0.6236	0.6281
ConsumerCyclical	0.7379	0.6087	0.7463	0.5845	0.6073
ConsumerNonCyclical	0.8704	0.6530	0.8753	0.7227	0.7285
Energy	0.8685	0.7701	0.8682	0.8083	0.8520
Financial	0.8002	0.6619	0.8067	0.5566	0.6238
Healthcare	0.8890	0.6918	0.8898	0.7652	0.8142
Services	0.7966	0.6035	0.8124	0.5823	0.6031
Technology	0.8378	0.6785	0.8427	0.7146	0.7294
Transportation	0.9306	0.7325	0.9307	0.8406	0.8825
Utilities	0.9103	0.7982	0.9096	0.8841	0.8924
Average	***0.8173***	***0.6715***	***0.8222***	***0.6860***	***0.7132***

Table 1: Area under curve (AUC) for all scoring methods

	error reduction			
Sector	s_{end}	s_{cnd}	d_{si}	d_{efigf}
BasicMaterials	0.0080	0.2072	0.2966	0.2411
CapitalGoods	0.0090	0.0520	0.3250	0.2948
Conglomerates	0.0550	0.5081	0.3881	0.3807
ConsumerCyclical	0.0322	0.3517	0.3895	0.3540
ConsumerNonCyclical	0.0382	0.6407	0.5503	0.5408
Energy	-0.0028	0.4267	0.3122	0.1092
Financial	0.0327	0.4283	0.5642	0.4863
Healthcare	0.0068	0.6423	0.5305	0.4066
Services	0.0778	0.5268	0.5508	0.5274
Technology	0.0303	0.5106	0.4489	0.4186
Transportation	0.0007	0.7409	0.5653	0.4101
Utilities	-0.0073	0.5520	0.2201	0.1600
Average	***0.0234***	***0.4656***	***0.4285***	***0.3608***

Table 2: Relative error reductions for s_{wend} over other methods

Notice the curious shape of the ROC curves in Figure 2: rather than having smoothly decreasing slopes (for ROC curves the slope corresponds to the class-membership likelihood ratio), after a certain point the slope is constant (to (1,1)). This is an indication that s_{wend} is giving equal (low) scores to a large number of entities. Examining the scores we see that, indeed, the direct method is giving scores of zero to many entities.[3]

s_{wend}=0 means that the entity is not linked to any (background) members of the class. This may largely be due to our limited data sample. A larger sample would contain (i) many more links and perhaps (ii) many more labeled background companies. Moreover, comparing different direct scores on these data obscures their differences, because (as is evident in Figure 2) due to the large number of zeros, for a given industry the AUCs cannot be very different for different direct scorings (which would correspond only to different slopes of the already-very-steep initial rise). By definition, on the cases with no links to background class members, all of the direct methods give zero scores.

Therefore, to assess the potential of the scores with more data, and to compare different direct scores on those cases where they *can* differ, we magnify the far-left part of the curves by looking only at those cases with at least one link to a background member of the class (i.e., ignoring the zero scores). The resultant ROC curves for s_{wend} are shown in Figure 3.

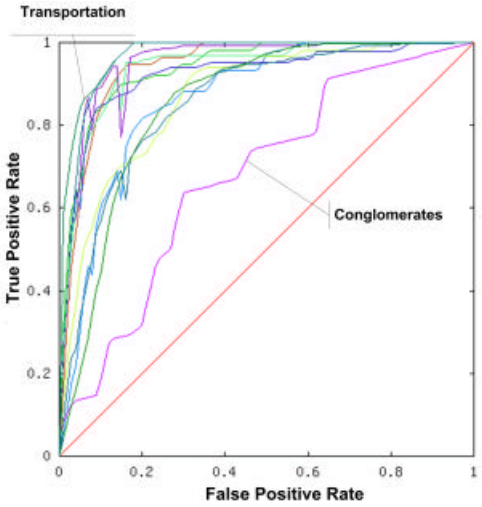


Figure 3: ROC curve for weighted, entity-normalized method, ignoring non-linked entities (averaged over 10 runs)

In Figure 3, most of the AUCs are 0.9 or better, and only one (Conglomerates, AUC=0.67) is less than 0.8. This demonstrates that s_{wend} can separate the entities by class remarkably well, in cases where it has a chance—i.e.,

[3] Giving scores of zero to entities *not* in the class is of course desirable. The problem here is that members of the class are receiving scores of zero. The percentage varies from sector to sector, and can be estimated by (one minus) the TP rate at the beginning of the final linear segment of the ROC curve. E.g., for Transportation approximately 10% of the members of the class receive zeros. For Capital Goods, approximately 50% receive zeros.

where there is at least one link to a known member of the class.

	area under curve (no zeros)			
Sector	s_{end}	s_{wend}	d_{si}	d_{efigf}
BasicMaterials	0.9106	0.9286	0.6442	0.6685
CapitalGoods	0.8321	0.8574	0.5299	0.5676
Conglomerates	0.5755	0.6668	0.7079	0.7169
ConsumerCyclical	0.8205	0.8602	0.5853	0.6107
ConsumerNonCyclical	0.9079	0.9317	0.7482	0.7578
Energy	0.9291	0.9281	0.8283	0.8522
Financial	0.8892	0.9107	0.6243	0.6646
Healthcare	0.9397	0.9405	0.7599	0.8078
Services	0.8143	0.8462	0.5712	0.5970
Technology	0.8373	0.8446	0.7051	0.7195
Transportation	0.9567	0.9624	0.8551	0.9124
Utilities	0.9397	0.9518	0.9076	0.9225
Average	***0.8627***	***0.8857***	***0.7056***	***0.7331***

Table 3: Area under curve (AUC) for all scoring methods ignoring non-linked entities

Table 3 reports the AUCs of all 5 scoring functions for the 12 classification tasks for this task. In most cases all the scoring methods classify considerably better than random (represented by the diagonal in ROC space), but again s_{end} and s_{wend} perform the best. The wend score consistently performs better than the other scores (with only a few exceptions). Table 4 shows the relative error reduction of the s_{wend} over the other methods. Even over s_{end}, it achieves a 15% error reduction on average.

	error reduction (no zeros)		
Sector	s_{end}	d_{si}	d_{efigf}
BasicMaterials	0.2019	0.7994	0.7846
CapitalGoods	0.1506	0.6966	0.6701
Conglomerates	0.2152	-0.1406	-0.1768
ConsumerCyclical	0.2209	0.6628	0.6408
ConsumerNonCyclical	0.2586	0.7290	0.7182
Energy	-0.0152	0.5810	0.5132
Financial	0.1945	0.7624	0.7339
Healthcare	0.0133	0.7521	0.6904
Services	0.1716	0.6413	0.6183
Technology	0.0444	0.4729	0.4458
Transportation	0.1298	0.7402	0.5702
Utilities	0.1994	0.4779	0.3777
Average	***0.1487***	***0.5979***	***0.5489***

Table 4: Relative error reductions for s_{wend} over other methods ignoring non-linked entities

It is important to emphasize that we are not claiming that these results show that s_{wend} is generally preferable. This will be domain and task dependent. For this particular domain, s_{wend} seems to be the better score. This general result is reinforced by examining the results on the finer-grained industry (rather than sector) affiliations. For 34 of the 97 industries the two methods produce identical generalization performance.[4] For the remaining 63 industries, s_{end} is superior for 11 and s_{wend} for 52. Figure 4 plots the AUCs of s_{wend} (vertical axis) and s_{end} (horizontal axis). Points above the diagonal indicate that s_{wend} has a higher AUC than s_{end}. Clearly, s_{wend} is the better performer on these finer-grained classification tasks, sometimes by a large margin.

Returning to the zero scores, the direct RVS method does not stand a chance when there are no links to a known member of the class. The indirect method is not so limited—the only time it will give a non-zero score for a class is if the entity in question is not linked to anything that a known member is linked to. Scoring all the companies with the indirect method indeed produces few zeros. Unfortunately (as shown in Table 1), the classification performance is not nearly as strong with the indirect methods. The indirect methods show a much wider range of performance, from Utilities (almost as good as with the direct score) down to Capital Goods (apparently random).

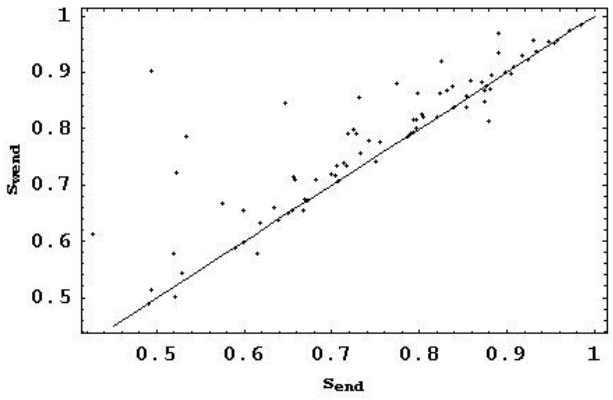


Figure 4. AUC of s_{wend} vs. AUC of s_{end} on the 97 industries

4.2 Hybrid methods

In order to improve the direct methods' performance on entities with no direct links to the class, it is possible to combine the direct and indirect methods, using the latter only when the former returns a zero.

Definition: The *weighted, efigf combined score* of an entity is:

$$s'(e,i) = d_{efigf}(e,i) * \min_k (s_{wend}(e,k))$$

$$cs\ (e,i) = \begin{cases} s_{wend}(e,i) \\ s'(e,i), if\ s_{wend}(e,i) = 0 \end{cases}$$

Thus, we use the weighted, entity normalized direct score s_{wend}, unless s_{wend} is zero, in which case we scale the efigf-score by the minimal, greater-than-zero s_{wend} to fit the d_{efigf}'s below the true weighted, entity normalized scores.

[4] For sparser data the two methods' scorings will become more similar—and exactly identical scorings are not necessary to produce identical ROC curves.

Using this approach, we see a modest improvement. On average we see 4% additional error reduction over s_{wend} (see Table 5). However, there are certain cases where additional error reduction is very large (Transportation, Energy error reduction >20%), and three cases where it increases error (on average 9% relative increase). This illustrates the need for a flexible framework within which a variety of RVS methods can be defined and tested.

Another approach to address the scoring of entities with no links to a known member of the class in question is to investigate degree-2 links (links to entities "two hops" away). Redefining the links in the direct RVS model results in a score, which is analogous to s_{end}, the simple entity-normalized direct RVS score, but follows links of degree two. Consider **w?** to be the analogue to **w,** except with two-hop links.

Definition: An entity e can be described by an *simple second-degree entity vector* $\mathbf{w?_j} = (w?_{j,1}, w?_{j,2}, \ldots)$, where:

$$w?_{j,k} = 1 \text{ if } w_{j,i} * w_{i,k} = 1 \text{ for any } e_i, e_k \text{ in } E$$

Definition: The *second-degree class-normalized direct RVS score* of entity e for class i is the inner product of $\hat{\mathbf{w}}''$ and the canonical class vector $\mathbf{c_i}$, normalized by the L1 norm of **c**.

$$s''_{cnd}(e,i) = \frac{\hat{\mathbf{w}}'' \cdot \mathbf{c}_i}{\sum c_j}$$

Again we can define a combined score:

Definition: The *weighted, second degree class-normalized combined score* of an entity is:

$$s''(e,i) = s''_{cnd}(e,i) * \min_k (s_{wend}(e,k))$$

$$cs''(e,i) = \begin{cases} s_{wend}(e,i) \\ s''(e,i), if\ s_{wend}(e,i) = 0 \end{cases}$$

	area under curve			rel. error red.	
Sector	s_{wend}	***cs***	***cs"***	***cs***	***cs"***
BasicMaterials	0.7339	0.7313	0.7677	-0.0098	0.1270
CapitalGoods	0.6810	0.6525	0.7187	-0.0891	0.1183
Conglomerates	0.7697	0.7702	0.7232	0.0024	-0.2019
ConsumerCyclical	0.7463	0.7178	0.7682	-0.1126	0.0862
ConsumerNonCyclical	0.8753	0.8859	0.8726	0.0850	-0.0215
Energy	0.8682	0.8981	0.9078	0.2267	0.3003
Financial	0.8067	0.7938	0.8129	-0.0671	0.0319
Healthcare	0.8898	0.8945	0.9136	0.0425	0.2163
Services	0.8124	0.8150	0.8234	0.0137	0.0586
Technology	0.8427	0.8458	0.8496	0.0200	0.0437
Transportation	0.9307	0.9470	0.9458	0.2347	0.2177
Utilities	0.9096	0.9185	0.9187	0.0979	0.1011
Average	*0.8222*	*0.8225*	*0.8352*	*0.0370*	*0.0898*

Table 5: AUC and relative error reduction with combined methods

As Table 5 shows this method improves further over s_{wend}. On average we get 9% relative error reduction with some reductions going up to 30% (for energy) and two additional being higher than 20% (Healthcare and Technology). Like with the weighted, efigf combined score cs, however, some sectors have an error increase, the largest being Conglomerates with 20%. (NB: by its nature, Conglomerates is the one sector for which we would not expect members to be linked to each other.) This illustrates that even in a domain where simple scores perform very well, more-complex scores can add value.

4.3 Comparing scores across sectors

The ROC analysis above evaluates the problem: given a sector, how well can companies be separated into those in the sector and those not. More specifically, it evaluates the scoring function's ability to rank the companies by probability of class membership. The dual question is: given a company, how accurately can it be placed into the "correct" sector?

The base rate for this classification problem will be the marginal probability of the most common class: in our data, 0.29 (Technology). The accuracy of s_{wend} for classifying companies into the correct sector was 0.68. Table 6 shows the accuracy for the companies in each sector. For only one sector (Conglomerates) was the classification accuracy worse than the base rate (0.15) and this sector also had the smallest number of members (recall that s_{wend} does not normalize for the size of the class). Classification is one (important) case where comparing scores across sectors is necessary. We will return to this in the follow-up analysis below.

Sector	Correct	Total	Accuracy
Technology	392	505	0.78
Energy	54	71	0.76
Transportation	28	38	0.74
Healthcare	131	180	0.73
Utilities	21	30	0.70
Financial	111	170	0.65
Services	286	444	0.64
ConsumerNonCyclical	38	60	0.63
BasicMaterials	47	104	0.45
ConsumerCyclical	36	99	0.36
CapitalGoods	17	73	0.23
Conglomerates	3	14	0.21
Overall	**1164**	**1788**	**0.65**
base rate (Technology)			*0.28*

Table 6: Accuracy for classifying companies in each sector

4.4 Other methods

How good are these results, with respect to other methods of company-affiliation classification? Our goal in this paper was to demonstrate the RVS model, and not to assess what is the best method for company affiliation identification. Nevertheless, for completeness we address this question briefly.

Running the relational learning program FOIL [10] on these data failed completely, returning a single clause for each company. We modified FOIL to search for more general theories, and it still performed far worse than the RVS methods. In retrospect, this is not surprising because FOIL (and many other ILP [5] algorithms) do not perform

numeric aggregations without having them be defined explicitly. The RVS scores may provide useful constructed features for ILP programs.

We created an ensemble, multi-document, full-text classification method, using the stories from which the links were extracted. This method performed similarly to s_{wend} but was two orders of magnitude slower. Interestingly, when the sector-specific word models were examined, the names of major companies in the sector were given high scores. So the text-based method chose to use these "links" in its own vector-space model.

In the financial literature and industry, companies are clustered into industry groupings based on correlations in their financial time series (and singular-value decompositions) [11]. Our experiments so far with these methods have not yielded remarkable performance on our classification tasks.

Probabilistic and statistically oriented relational learning methods, such as PRMs [12], and relational versions of naïve Bayes [13], decision trees [14], etc., hold the most promise for competing with the RVS model. These methods do perform aggregations over the values of the attributes at linked nodes. In particular, properly utilized (weighted) COUNT or MODE operations would incorporate the fundamentals of the basic, direct RVS scores. However, even if they performed competitively, they far more complex learning procedures than the RVS scoring functions.

5. Discussion and Followup

So, what does our case study illustrate about the relational vector-space model? First, it shows that there are domains where the interlinkage between class members is strong enough for simple scoring methods based only on linkage to capture much of the "signal" needed for good classification. And for some tasks the scoring can lead to remarkable classification accuracy. For example, even though Transportation companies represent only 2% of the companies, the excellent Transportation scores (AUC>0.9) lead to a classification accuracy of 74%, when classifying by choosing the highest sector-score (of the 12).

Intuitively, we expect the direct RVS methods to excel when (as in Figure 1) entities are more likely to be linked to other entities with the same class membership. This intuitive notion is captured more formally by *relational autocorrelation* [15]: the correlation between values of the same attribute on linked entities "represents an extremely important type of knowledge about relational data, one that is just beginning to be explored and exploited for learning statistical models from relational data" (ibid). We can use this notion to understand the RVS model in more detail.

Adapting Jensen & Neville's [15] definition to our context, consider a set of entities E, an attribute *f*, and a set of paths P that connect objects in E.

Definition: Relational autocorrelation C' is the correlation between all pairs $(f(x_1),f(x_2))$ where $x_1, x_2 \in E, x_1 \neq x_2$ and such that $\exists p(x_1, x_2) \in P$.

Let us define *degree-k* relational autocorrelation as further restricting the length of $p(x_1, x_2)$ to be *k*. Intuitively, the direct RVS method should be appropriate when the *degree-1* relational autocorrelation in the entities' class values is high ("homophily"). We can use an existing measure of relational autocorrelation to verify this. Following Jensen & Neville we use Pearson's corrected contingency coefficient to measure class-value autocorrelation.

For our sector-classification problem, the degree-1 relational autocorrelation considering all classes is 0.84, reflecting our intuition from inspecting Figure 1. Figure 5 shows for each class the classification performance (accuracy) plotted against the class vs. not-class degree-1 autocorrelations. The rankings of performance and autocorrelation are very similar (Pearson's correlation coefficient is 0.76). This high value is due to a large part to Conglomerates, which has the lowest autocorrelation and the lowest accuracy. Nonetheless it suggests that the performance of the direct RVS method indeed is related to the degree-1 relational autocorrelation in the class values.

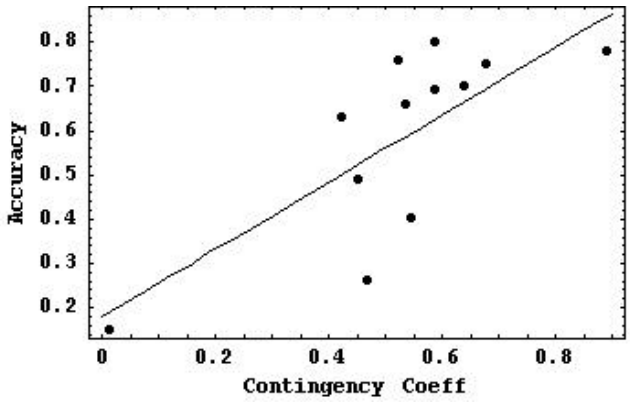


Figure 5: Accuracy versus degree-1 autocorrelation

More specifically, the direct RVS score itself is a measure of degree-2 relational autocorrelation where the path $p(x_1,x_2)$ passes through the entity to be classified. If the degree-1 relational autocorrelation is high, one would expect entities connected by paths of length 2 through an entity of class C, also to have class C (this is the condition for the direct RVS score to be effective for classification).

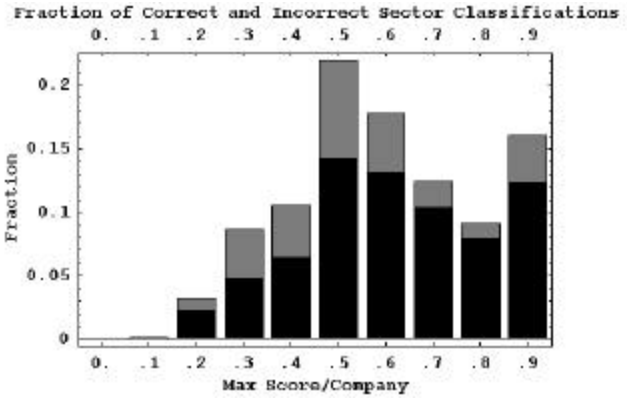


Figure 6: Fraction of correct and incorrect Sector Classifications (black are correct classifications, gray are incorrect classifications)

This suggests that the RVS scores can be used for assessments of the nature of the relational autocorrelation in a graph, that are finer-grained than given by the contingency coefficient. For example, for our sector-classification problem, Figure 6 is a histogram, plotting the distribution of companies over the maximum of s_{wend} for any of the 12 classes. The black (gray) shading shows the percentage of companies with the same (different) class as the class with the maximum score. Interestingly, the distribution shows that for this domain, most (>75%) of the entities have a (weighted) majority of the links to entities of a single class. More often than not, this class is correct.

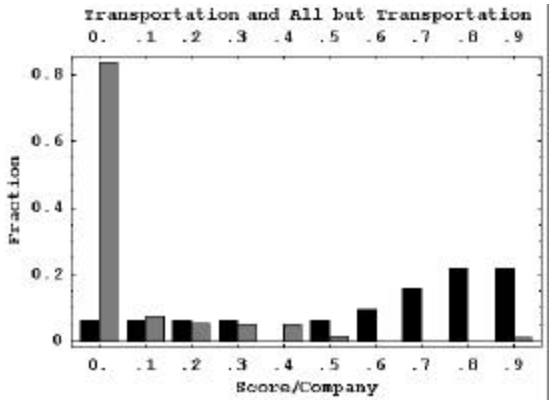


Figure 7: Sector specific s_{wend} scores for Transportation (gray is All but Transportation, black is Transportation)

Let us use s_{wend} to view two of the particular sector classification tasks, Transportation (high AUC & accuracy) and Capital Goods[5] (low AUC & accuracy). Figure 7 and Figure 8 show histograms of the sector-specific s_{wend} scores for the members of the class (black) and the non-members (gray). We can see clearly that Transportation companies are primarily linked to other Transportation companies, and other companies are not. Capital Goods companies, on the other hand, show very different connectivity—they are not primarily linked to other Capital Goods companies. In fact, their linkage to other Capital Goods companies is remarkably similar to that of the rest of the companies.

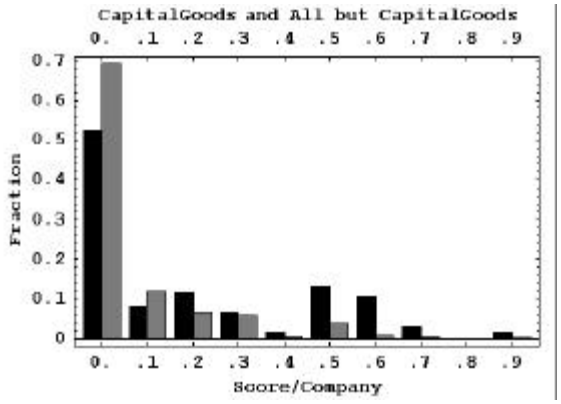


Figure 8: Sector specific s_{wend} scores for Captial Goods (gray is All but CapitalGoods, black is CapitalGoods)

Finally, consider the comprehensive view of class-interlinkage given in Figure 9 (on the last page), which shows the class interlinkage for all class pairs. Each individual graph shows the averages across the members of the class of the s_{wend} scores for each of the 12 classes. This figure gives a condensed visualization of the class-specific interlinkage in the graph.

We argue that this visualization could lead to insights about the classes. Pretend for the moment that we did not already have a basic understanding of the sectors. We see that Capital Goods has high linkage to most of the other classes. Transportation, on the other hand is linked primarily with itself.[6] And Services are linked almost uniformly to the rest of the sectors. Utilities are linked to Energy and Transportation (and in contrast to the rest of the sectors, not to Technology much at all). Each of these properties makes good sense for the corresponding class.

6. LIMITATIONS AND FUTURE WORK

For this study we limited ourselves to relatively simple RVS scoring functions. This was partially due to our desire to flesh out the basics of the model first before getting fancy, but more due to the remarkable performance of the basic methods in our case-study domain.

The RVS scoring functions are "learning" procedures only in the sense that nearest-neighbor classifiers are: they simply apply a scoring function to a database of instances-- no feature selection or parameter estimation takes

[5] Conglomerates is similar, but has only 13 member companies (as compared to 61 for Capital Goods).

[6] We have not normalized here by the size of the class here, in keeping with the rest of the paper (so Technology is weighted heavily across most of the classes). Doing so gives a different, and equally intriguing visualization.

place. Indeed, s_{wend} could be considered a "Relational Neighbor" classifier [16], that takes advantage of class homophily. Provost et al. argue that such a simple model should generally be used as a baseline for more complicated approaches, because it seems to perform remarkably well in many domains [16]. Jensen & Neville found high relational autocorrelation for almost all attributes they examined in linked movie data [15]. Furthermore, homophily has been observed in human groups with respect to a wide variety of descriptive variables, and is one of the basic premises of theories of social structure [17]. Chakrabarti et al. take advantage of autocorrelation in class values to classify hypertext documents [18]. Their procedure learns a probabilistic model based on the classes of related entities, and therefore can capture more complex relationships than simply homophily.

There are several ways in which the current model is limited. We only consider a single link type. This does not restrict the model's applicability, because (as we did in our case study) the type of links can simply be ignored. However, it may obscure information that is important for classification. The model as presented could be extended to handle multiple link types simply by creating multiple vectors (one per link type) and concatenating them. Alternatively, different models could be produced for different link types, and selected among or applied as an ensemble. Whether or not these would be effective techniques is a subject for future study.

We also only consider a single entity type. This is a more fundamental limitation of the model, and we have not considered carefully how to extend it. One obvious way to apply the model to data with multiple types of entities is to focus on one entity type, and consider paths between these entities (perhaps going through other entities) to be the links.

The direct RVS scores (as presented) abstract away most of the graph structure, only considering adjacency. This is the source of the model's elegant simplicity, but it also limits the types of problems on which it will be effective. It could be extended by defining links in the model to be paths of length greater than one. These could be treated similarly to multiple link types, as discussed above.

We have assumed that more data will (partially) resolve the issue with many zero scores (described in Section 4.1). We have little support for this assumption, but it seems reasonable. We have procured another data set to test with; however, we have not yet completed the data preprocessing necessary to make the two data sets comparable.

Finally, we have looked at different sector and industry classifications (SIC codes and Hoover's classification) with qualitatively similar results, but have not studied them comprehensively. We would like to show that the RVS model with newswire-extracted links can model various, different classifications that have little similarity to each other (the aforementioned surprisingly do not) but are nevertheless meaningful.

7. CONCLUSIONS

The relational vector-space model is a useful abstract representation for studying relational classification. With simple choices for its components (entity vector, class vector, normalization function) it represents intuitive notions of classification by relational autocorrelation. With more complicated choices, it can represent more complex classification models on linked data (still abstracting away much of the graph structure).

In our case study of company affiliation classification, relatively simple scoring functions performed remarkably well, illustrating the potential utility of the RVS model. However, the RVS scores may be most useful as feature constructors in other, more complicated systems. Relational learners can include these scores as (additional) aggregation functions. Standard feature-vector learners can use the RVS scores to take into account an important part of relational structure.

The case study also illustrated the advantage of the structure that the RVS model places on the space of scoring functions, allowing them to be explored systematically. Although the improvement for this domain was not dramatic, the results of combining the different scores do suggest that combined RVS scoring models may be advantageous in certain domains.

8. ACKNOWLEDGMENTS

We thank Claudia Perlich, Shawndra Hill, David Jensen, Tom Fawcett, and Roger Stein for discussions during various stages of this project. This work is sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement F30602-01-2-585.

9. REFERENCES

[1] S. Dzeroski and N. Lavrac, *Relational data mining*. Berlin; New York: Springer, 2001.

[2] MRDM Workshop on Multi-Relational Data Mining at KDD, Edmonton, Alberta, Canada, July 23, 2002

[3] L. Getoor and D. Jensen, "Learning Statistical Models from Relational Data - AAAI 2000 Workshop," American Association for Artificial Intelligence (AAAI), Menlo Park, California, Technical Report WS-00-006, 2000.

[4] G. Salton and M. J. McGill, *Introduction to modern information retrieval.* New York: McGraw-Hill, 1983.

[5] S. Muggleton, *Inductive logic programming*. London: Academic Press in association with Turing Institute Press, 1992.

[6] K. M. Kahle and R. A. Walking, "The Impact of Industry Classification on Financial Research," *The Journal of Financial and Quantitative Analysis*, vol. 31, pp. 309-335, 1996.

[7] J. Swets, "Measuring the Accuracy of Diagnostic Systems," *Science*, vol. 240, pp. 1285-1293, 1988.

[8] F. Provost and T. Fawcett, "Robust Classification for Imprecise Environments," *Machine Learning*, vol. 42, pp. 203-231, 2001.

[9] D. J. Hand, *Construction and Assessment of Classification Rules*. Chichester, UK: John Wiley and Sons, 1997.

[10] J. R. Quinlan and R. M. Cameron-Jones, "FOIL: A midterm report," 6th European Conference on Machine Learning, 1993.

[11] P. Gopikrishnan, B. Rosenow, V. Plerou, and H.E. Stanley, "Identifying business sectors from stock price fluctuations," Phys. Rev. E 64, 035106R, 2001.

[12] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, "Learning Probabilistic Relational Models," 16th International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, 1999.

[13] J. Neville, D. Jensen, B. Gallagher, and R. Fairgrieve, "Simple estimators for relational Bayesian classifiers," Department of Computer Science. University of Massachusetts Amherst., Amherst, MA, USA, Technical Report 03-04, 2003.

[14] D. Jensen and J. Neville, "Schemas and models," Multi-Relational Data Mining Workshop, 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining., 2002.

[15] D. Jensen and J. Neville, "Linkage and autocorrelation cause feature selection bias in relational learning," Nineteenth International Conference on Machine Learning (ICML2002), 2002.

[16] F. Provost, C. Perlich, and S. Macskassy, "Relational learning problems and simple models." IJCAI-2003 Workshop on Learning Statistical Models from Relational Data, 2003.

[17] P. Blau. *Inequality and Heterogeneity: A Primitive Theory of Social Structure*. New York: The Free Press, 1977.

[18] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks. ACM SIGMOD 1998.

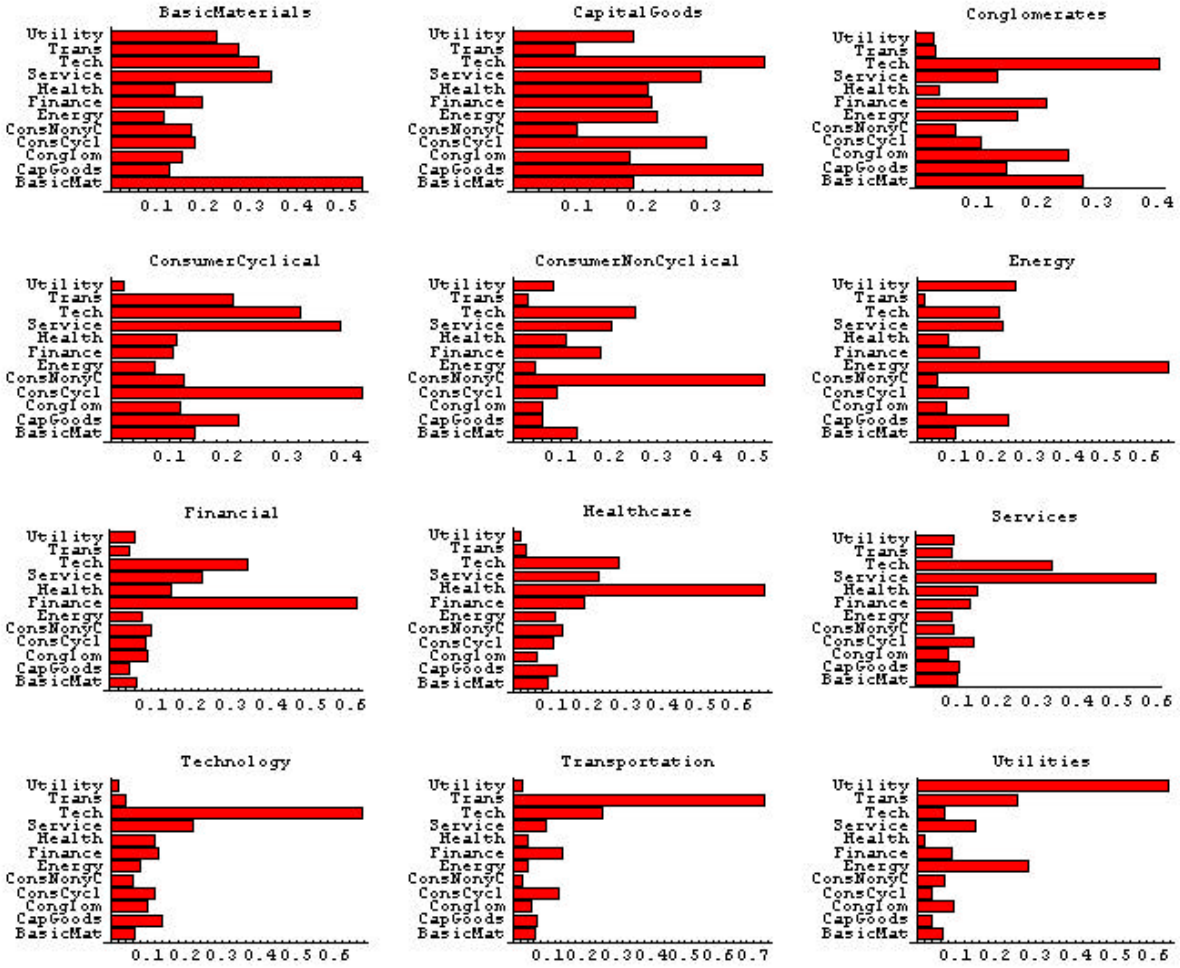


Figure 9. Average class-specific s_{wend} scores by class, as visualization of class interlinkage in graph

Categorizing Unsupervised Relational Learning Algorithms*

Hannah Blau **Amy McGovern**
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003-9264
{blau,amy}@cs.umass.edu

Abstract

We outline some criteria by which to compare unsupervised relational learning algorithms, and illustrate these criteria with reference to three examples: SUBDUE, relational association rules (WARMR), and Probabilistic Relational Models. For each algorithm we ask, What form of input data does it require? What form of output does it produce? Can the output be used to make predictions about unseen inputs? Categorizing the existing unsupervised relational learning algorithms helps us to understand how each algorithm relates to the others (no pun intended). We can identify important gaps in coverage that could be fruitful areas for future research.

1 What do we mean by *unsupervised*?

In this paper we outline some criteria by which to compare unsupervised relational learning algorithms. We begin by clarifying what we mean by an *unsupervised* learning algorithm. A *supervised* learning algorithm distinguishes one attribute of its input instances as the target and learns a model designed to predict the value of the target attribute for previously unseen inputs. The target attribute can be discrete, as in classification, or continuous. An *unsupervised* learning algorithm does not treat any particular attribute of its input instances as the target to be learned. There is no teacher who gives the correct answer; there *is* no one correct answer. In some cases, the model produced by an unsupervised learning algorithm can be used for prediction tasks even though it was not designed for such tasks. The distinction between supervised and unsupervised learning is a spectrum on which some algorithms are at the extremes and others are toward the middle. SUBDUE is clearly an unsupervised learning algorithm. It recognizes repeated substructures in a labeled graph, and can be used for graph compression [Cook and Holder, 1994] and for hierarchical clustering [Jonyer *et al.*, 2001], but not prediction. Relational Markov Networks [Taskar *et al.*, 2002] are designed for discriminative training: they fall at the supervised end of the spectrum. Probabilistic Relational Models are more toward the middle. PRMs learn a dependency structure which can enhance a domain expert's understanding of the data [Getoor *et al.*, 2001]. They can model uncertainty in the relational structure of the domain [Getoor *et al.*, 2002]. They can be used for classification and for clustering [Taskar *et al.*, 2001]. The underlying learning algorithm is the same, but the relational data structures given as input are adapted to the desired task.

*This effort is supported by DARPA and AFRL under contract numbers F30602-00-2-0597 and F30602-01-2-0566, and by NSF under contract number EIA9983215. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA, AFRL, NSF, or the U.S. Government.

2 Criteria of comparison

Unsupervised relational learning algorithms can be categorized along several different axes:

- What form of input data does the algorithm require?
- What form of output does it produce?
- Can the output be used to make predictions about unseen inputs?

To describe the input data configuration, we employ the terms *object*, *link*, and *attribute*. (We choose *link* instead of *relation* to avoid confusion with the terminology of relational database management systems.) In our framework, relational data consist of objects connected together by links. Both objects and links can have attributes. An attribute is a name-value pair.

The input for any learning algorithm that claims to be "relational" must have links as well as objects. A link can be represented explicitly by an edge in a graph, or implicitly by a pointer to the related object. The number of attributes allowed for each object or link can be none, exactly one, or many. The input database can consist of a single connected component, or a set of connected components.

The output of a relational learning algorithm is a pattern (using the term loosely) that expresses a generalization supported by the input data. The scale of the pattern might be a single object, or a structure consisting of a group of related objects and the links that connect them. All patterns produced by a relational learning algorithm are descriptive because they

capture regularities of the input data; some patterns can also be used to make predictions about unseen data.

Categorizing the existing unsupervised relational learning algorithms helps us to understand how each algorithm relates to the others (no pun intended). Our goals in developing this categorization are

- to establish a common vocabulary in which to express the similarities and differences of relational learning algorithms;
- to identify interesting areas of unsupervised relational learning that are currently underdeveloped.

3 Three example algorithms

We illustrate our multi-dimensional categorization of unsupervised relational learning algorithms by comparing three systems that differ widely in their input and output formats.

The WARMR algorithm [Dehaspe *et al.*, 1998; Dehaspe and Toivonen, 2001] finds relational association rules or, to use the vocabulary of the authors, *query extensions*. The algorithm takes as input a Prolog database and a specification (in the WARMODE language) that limits the format of possible query extensions. The output of WARMR is a set of query extensions, all of which refer to the object designated as the *key* parameter. The query extensions are not limited to attributes of the key object, but can include its links to other objects and their attributes.

The SUBDUE system [Cook and Holder, 1994] iteratively discovers repeated substructures in a graph and compresses the graph by replacing the repeated substructure with a single vertex. The algorithm takes as input a labeled graph and a set of rules intended to bias the search process toward structures that are deemed more interesting. SUBDUE returns as output the substructure selected at each iteration as the best to compress the graph.

Probabilistic Relational Models (PRM) reinterpret Bayesian networks in a relational setting. PRMs have been evolving rapidly over the past few years; we focus here on the version described in [Getoor *et al.*, 2002]. A PRM captures the probabilistic dependence between the attributes of interrelated objects. It can also model uncertainty about the link structure. *Reference uncertainty* means we know how many links there are in the graph, but we don't know what their endpoints are. *Existence uncertainty* means we don't know how many links there are and have to consider the possibility that any pair of objects (of the appropriate types) might be linked. The input to the PRM learning algorithm is a database schema (specifying objects, links, and attributes) and an instantiation of that schema (a set of relational tables).

4 Input criterion of comparison

The first criterion of comparison concerns the input to the unsupervised relational learning algorithm. Our three example algorithms have very different data representations, but conceptually we can view their input in terms of objects and links. For SUBDUE the mapping is straightforward: objects correspond to vertices in the graph, and links to edges. SUBDUE requires exactly one attribute on each object and link in the graph: a label.

In the Inductive Logic Programming approach of WARMR, the input data are a set of Prolog facts, describing both objects and links. The predicate name is the equivalent of a type attribute. For example (from [Dehaspe and Toivonen, 2001, p. 191]), a fact such as

$$customer(allen).$$

represents an object of type **customer** with identifier **allen**. A fact such as

$$parent(allen, bill).$$

represents a link of type **parent** between the **allen** object and the **bill** object. The WARMR data model allows both objects and links to have multiple attributes besides type, which would be represented by additional arguments to the **customer** and **parent** predicates.

Our use of the terms "object" and "link" does not coincide with the terminology of [Getoor *et al.*, 2002]. What we call an object corresponds to the instantiation of an *entity class* in the PRM. What we call a link corresponds to the instantiation of a *relationship class*. The *reference slots* of the relationship class tell us the endpoints of the link. Both entity classes and relationship classes can have *descriptive attributes*, which we would simply call attributes. So any object or link in the PRM input can have multiple attributes. Could there be a link with no attributes? No. Even if the relationship class has only reference slots and no descriptive attributes, we still say that the link has one type attribute because in the PRM we know to what class this link belongs. For example, the PRM for the citation domain has a class representing the "cites" relationship between one paper and another. This is equivalent in our vocabulary to a link of type "cites" going from the citing paper to the cited paper. Keeping this translation of terminology in mind, we conclude that every object and link in the PRM's input has at least one attribute, its type, and possibly more.

5 Output criterion of comparison

The second criterion of comparison concerns the output produced by the unsupervised relational learning algorithm. Does the algorithm discover patterns at the level of individual objects, or at the level of subgraphs? (By "subgraph" we mean a structure containing at least one link with its associated objects.) SUBDUE searches for repeated substructures using an approximate graph match, and at each iteration returns the substructure which achieves the maximum graph compression when it is collapsed to a vertex. These are certainly patterns at the subgraph level. PRMs also discover patterns at the subgraph level. The result of training a PRM is an estimate of the joint probability distribution of attribute values (and link structure, in the case of reference or existence uncertainty) over the entire network.

Relational association rules are in a gray area. The WARMR algorithm requires that some predicate be designated as the key. All query extensions must contain the key predicate. For example, if **customer** is the key then all the rules will be about customers. (A link predicate such as **parent** can also be designated the key.) The association rules mention other objects to which the customer is linked, and the

attributes of those related objects. So all the discovered patterns concern the key object (or link) but can draw upon the relational neighborhood surrounding the key.

6 Predictive criterion of comparison

Generally the goal of an unsupervised learning algorithm is descriptive. We hope that the discovered patterns capture the essential regularities of the input dataset. However, for some algorithms it is possible to make predictions about new inputs based on the patterns observed in the training data. Relational association rules could be applied to make predictions about the key object (or link). As noted in Section 1, PRMs can be used for classification [Taskar *et al.*, 2001; Getoor *et al.*, 2002]. SUBDUE's output cannot be exploited for prediction. There is no reason to assume the substructure that provides maximum compression in one input graph would do the same in another graph.

7 Conclusion

We have presented one approach to categorizing unsupervised relational learning algorithms, and applied it to three examples. These same criteria of comparison would be relevant for other algorithms we have not discussed, such as frequent subgraph discovery [Kuramochi and Karypis, 2001], and stochastic link and group detection [Kubica *et al.*, 2002]. We aim to establish a common vocabulary in which we can compare systems that have very different input/output specifications. Categorizing the current algorithms helps us identify important gaps in unsupervised relational learning that could be fruitful areas for future research.

Acknowledgments

We are indebted to the discussions of the Structuring Data working group of the Knowledge Discovery Laboratory at the University of Massachusetts Amherst. Members of this group are Andrew Fast, Lisa Friedland, Michael Hay, David Jensen, and Jen Neville, all of U Mass Amherst, and Pat Riddle of the University of Auckland.

References

[Cook and Holder, 1994] Diane J. Cook and Lawrence B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255, 1994.

[Dehaspe and Toivonen, 2001] Luc Dehaspe and Hannu Toivonen. Discovery of relational association rules. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 189–212. Springer-Verlag, 2001.

[Dehaspe *et al.*, 1998] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, editors, *4th International Conference on Knowledge Discovery and Data Mining*, pages 30–36. AAAI Press., 1998.

[Getoor *et al.*, 2001] Lise Getoor, Nir Friedman, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*. Springer-Verlag, 2001.

[Getoor *et al.*, 2002] Lise Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2002.

[Jonyer *et al.*, 2001] Istvan Jonyer, Diane J. Cook, and Lawrence B. Holder. Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research*, 2:19–43, 2001.

[Kubica *et al.*, 2002] Jeremy Kubica, Andrew Moore, Jeff Schneider, and Yiming Yang. Stochastic link and group detection. In *The Eighteenth National Conference on Artificial Intelligence*, pages 798–804, Jul 2002.

[Kuramochi and Karypis, 2001] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320, 2001.

[Taskar *et al.*, 2001] Benjamin Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In Bernhard Nebel, editor, *Proceeding of IJCAI-01, 17th International Joint Conference on Artificial Intelligence*, pages 870–878, Seattle, US, 2001.

[Taskar *et al.*, 2002] Benjamin Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*, Edmonton, Canada, 2002.

Aggregation versus selection bias, and relational neural networks

Hendrik Blockeel and Maurice Bruynooghe
Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium

Abstract

Current relational learners handle sets either by aggregating over them or by selecting specific elements, but do not combine both. This imposes a significant, possibly undesirable bias on these learners. We discuss this bias, as well as some ideas on how to lift it. In the process, we introduce the notion of relational neural networks.

1 Biases of Relational Learners

Among the many approaches to relational model learning that currently exist, a distinction can be made with respect to how they handle one-to-many and many-to-many relations, or, equivalently, how they handle sets of objects.

To illustrate this, consider a database with just a single relation "Person" with attributes Mother, Father, and Sex. (Mother and Father are foreign keys to Person.) and consider the following simple concepts:

A. people who have two children

B. people who have a son (that is, at least one)

C. people who have two sons

In all three cases, we want to classify persons, based on properties of (a set of) persons related to them. The parentheses around "a set of" indicate the two different kinds of approaches that we distinguish here, a distinction also mentioned by Jensen and Neville (2002).

The first kind of relational methods use aggregate functions to handle sets. The result of an aggregate function, obviously, is a property of the set as a whole, not of individual elements of the set. Among these methods we count, e.g., probabilistic relational models (PRMs) (Getoor et al., 2001) , or "propositionalization" approaches that include aggregates, such as the one by Krogel and Wrobel (2001).

A second kind of relational methods handles sets by looking at properties of their elements. Typically, tests are of the form "there exists an x in the set such that P(x) holds", with P a relatively complicated condition. Most inductive logic programming (ILP) systems follow this approach.

Let us call methods of the first kind, aggregating methods; and methods of the second kind, selective methods (in the sense that they select an element from the set and investigate properties of that single element). Referring to the example concepts above, we can then state that aggregating methods can easily express A, but not B, whereas selective methods can easily express B, but not A. Importantly, *none of the approaches mentioned can easily express concept C*, because this description contains both selection and aggregation (select all male children, and count only these).

More formally, if we express class definitions in the relational algebra and write them as $\sigma_{C_1}(\mathcal{F}(\sigma_{C_2}(R)))$ with R the result of joining the original relation with a relation it links to, then selective methods such as ILP focus on the construction of C_2 and fix C_1 and $\mathcal{F}$ to denote existence (count > 0), whereas aggregating methods focus on constructing a good C_1 and $\mathcal{F}$ but fix C_2 to be true.

For instance, PRMs, as defined by Getoor et al. (2001) cannot learn concept C without having separate relations for sons and daughters. Manually introducing these separate relations of course presupposes that the user is aware of the possible importance of these concepts. Alternatively, one could define a large number of aggregate functions that have appropriate selection conditions built in; in that case, a search through a space of aggregate functions is needed.

In an ILP setting, one could of course define aggregate functions as background knowledge. Then, e.g., the rule `p(X) :- count(Y, (child(X,Y), male(Y)), 2)` expresses concept C. The main difficulty here is that the second argument of the count meta-predicate is itself a query that is the result of a search through some hypothesis space. It is not obvious how such a search should be conducted; the many results in ILP on how to search a first-order hypothesis space efficiently (Nienhuys-Cheng and De Wolf, 1997) do not consider the case where the resulting hypothesis will be used as the argument of a metapredicate.

ILP-like approaches that do not include aggregate functions, can still express concept C as, e.g., "the person has a male child x and a male child y and $x \neq y$ and there does not exist a child z such that z is male and $z \neq x$ and $z \neq y$"; but in practice, the length of this rule, as well as the occurrence of a negation (the scope of which is again a conjunction of multiple literals) make it difficult to learn, and of course also the comprehensibility of the result is negatively influenced.

To our knowledge no currently existing approaches can construct theories that combine aggregate functions with (reasonably complex) selections on the set to be aggregated.

2 Combining Aggregation with Selection

In databases, both aggregation and selection are very natural operations, and ideally a relational learning system should be able to combine both in the models it builds. In order to achieve this goal, it is necessary to define a search space of hypotheses that combine aggregations and selections, and find a more or less efficient way to navigate through this search space. This is currently an open problem. We here list a number of ideas that could be investigated further. We divide them into symbolic and subsymbolic approaches.

2.1 Symbolic Approaches

To build a concept in symbolic form, a search space has to be traversed that consists of combinations of aggregations and selections. This could be done in a hill-climbing way, but it appears that in some cases the search can be made slightly more exhaustive without increasing its computational complexity much. For instance, counting the number of children of a person takes just as much work as counting the number of sons and daughters separately, and a simple addition of these counts yields the total number of children. More generally, given a partition $\{S_1, \ldots, S_n\}$ of a set S, aggregates of S can often be computed efficiently from aggregates of the S_i, and the latter can all together be computed as efficiently as computing the aggregate for S. This holds at least for the often occurring aggregate functions count, sum, average, min, max. Thus, when we search for conditions of the form $\mathcal{F}(\sigma(S))\theta c$ with $\mathcal{F}$ an aggregate function, σ some selection, and θ some operator $(\leq, =, \ldots)$, a certain subspace of all possible σ's can be searched exhaustively at very little additional computational cost, compared to considering only the condition $\mathcal{F}(S)\theta c$. This suggests a straightforward possible improvement to some of the existing approaches.

2.2 Subsymbolic Approaches

Another direction for future research that seems interesting, is that of modelling relational databases with neural networks. Neural networks are usually considered propositional learners. A number of approaches exist to extend them to the context of first order logic, but not (to our knowledge) to that of relational databases, which could in fact be simpler. One approach to do that is based on the following observation.

Any data can be modelled using only two basic data structures: tuples and sets. (The relational data model is based on just these two notions.) Propositional learning algorithms handle tuples; to make them relational, it is sufficient to add the capability to process sets. (This is consistent with De Raedt (1998), who identifies multi-instance learning as the simplest "relational" learning task; it is indeed the simplest case where a single example is described by a set of tuples.)

The input of a standard feedforward neural network is a tuple. A relational neural network should in addition have the ability to handle sets, which can have an unlimited number of unordered elements. Recurrent neural networks have this capability: by feeding the output of a layer back into the network, they can aggregate information over an indefinite number of previous inputs. They are typically used for tasks such as time series prediction, where an input at time t can influence the output at time $t + k$ with k not bounded, but they can just as well be used for processing sets.

Thus, a relational neural network would essentially consist of "normal" and "aggregating" nodes; an aggregating node is simply a node that is fed back into a lower layer. Such a relational neural network would have the same structure as the skeletons used in PRMs. Where the PRM skeleton contains an aggregate function, the relational neural net contains one or more aggregating nodes. Relational neural nets are very similar to Ramon, Driessens and Demoen's (2002) "neural logic programs", with as main difference that Ramon et al. consider fixed combination functions for the different kinds of nodes and handle sets using nodes with a variable number of inputs, instead of recurrent nodes.

Relational neural networks would have as advantage over the other approaches that they can *learn* an aggregate function, without that function being pre-encoded in the network, and with selection possibly integrated in it. Thus, training the relational neural network automatically constitutes a search through aggregations and selections simultaneously. Moreover, a wider variety of aggregate functions is considered: not just sums, counts, etc. but also more exotic functions. On the other hand, the learnability of the relational neural networks we propose here, is an open problem. It is known that recurrent neural networks are harder to train than feedforward networks. Increasing the number of layers, as we do here, may further decrease learnability. We believe these issues are worth further investigation.

Acknowledgement

The first author is a postdoctoral fellow of the Fund for Scientific Research of Flanders, Belgium.

References

[1] L. De Raedt. Attribute-value learning versus inductive logic programming: the missing links (extended abstract). In D. Page, editor, *Proc. 8th Int'l Conf. on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 1–8. Springer, 1998.

[2] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In S. Džeroski and N. Lavrac, editors, *Relational Data Mining*, pages 7–34. Springer-Verlag, 2001.

[3] D. Jensen and J. Neville. Schemas and models. In *Proc. of the ACM SIGKDD 2002 Workshop on Multi-Relational Data Mining (MRDM 2002)*, pages 56–70, 2002.

[4] M.-A. Krogel and S. Wrobel. Transformation-based learning using multi-relational aggregation. In *Proc. 11th Int'l Conf. on Inductive Logic Programming*, pages 142–155, 2001.

[5] S.-H. Nienhuys-Cheng and R. De Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence*. Springer, 1997.

[6] J. Ramon, K. Driessens, and B. Demoen. Neural logic programs. Unpublished, 2002.

Feature Extraction Languages for Propositionalized Relational Learning

Chad Cumby **Dan Roth**
Department of Computer Science
University of Illinois, Urbana, IL 61801
{*cumby,danr*}*@uiuc.edu*

Abstract

We study representations and relational learning over structured domains within a propositionalization framework that decouples feature construction and model construction.

We describe two complementary approaches that address three aspects of the problem: First, we develop and study a flexible knowledge representation for structured data, with an associated language that provides the syntax and a well defined equivalent semantics for expressing complex structured data succinctly. Second, we use this language to automate the process of feature construction by expressing 'types' of objects in the language, which are instantiated in the ground data, allowing us to determine the level at which learning is done. Finally, this process of re-representation of the domain allows general purpose learning schemes, such as feature efficient linear algorithms and probabilistic representations, to be defined over the resulting space, yielding efficient and expressive learning of relational functions over a structured domain using propositional means.

1 Introduction

In a variety of AI problems, such as natural language understanding related tasks and visual inference, there is a need to learn, represent and reason with respect to definitions over structured and relational data. Examples include learning to identify properties of text fragments such as functional phrases and named entities, identifying relations such as "A is the assassin of B" in text, learning to classify molecules for mutagenicity from atom-bond data in drug design, learning to identify 3D objects in their natural surrounding and learning a policy to map goals to actions in planning domains.

In all these cases it is necessary (1) to represent and reason with *structured* domain elements in the sense that their internal (hierarchical) structure can be encoded, and learning functions in these terms can be supported, and (2) it is essential to represent concepts and functions *relationally*, in the sense that different data instantiations may be abstracted to yield the same representation – so that evaluation of functions over different instantiations will produce the same output.

The challenge is to provide the expressivity necessary to deal with large scale and highly structured domains such as natural language and visual inference and at the same time meet the strong tractability requirements for these tasks. Propositional representations might be too large, could lose much of the inherent domain structure and consequently might not generalize well. This realization has renewed the interest in studying relational representations both in the knowledge representation and reasoning (KRR) community and in the learning community. As it turns out, both are relevant to our approach. The main effort in the knowledge representation and reasoning community has been to identify classes of representations that are expressive enough to allow reasoning in complex situations yet are limited enough as to support reasoning efficiently [Levesque and Brachman, 1985; Selman, 1990]. It has become clear that propositional representations are not sufficient, and effort has been devoted to studying languages that are subsets of first order logic, such as description logics and frame representation systems [Borgida and Patel-Schneider, 1994], as well as probabilistic augmentations of those [Koller *et al.*, 1997].

The expressivity vs. tractability issue has been addressed also from the learning perspective, and a similar tradeoff has been observed and studied. While, in principle, Inductive Logic Programming (ILP) methods provide the natural approach to these tasks in that they allow induction over relational structures and unbounded data structures, theoretical and practical considerations render the use of unrestricted ILP methods impossible. These methods have also been augmented with the ability to handle uncertainty [Kersting and Raedt, 2000] although, as expected, this makes some of the computational issues more severe - studies in ILP suggest that unless the rule representation is severely restricted the learning problem is intractable [Muggleton and De Raedt, 1994; Dzeroski *et al.*, 1992; Cohen, 1995a; 1995b].

The main way out of these computational difficulties has been via the use of propositionalization methods that attempt to learn classifiers for relational predicates via propositional algorithms, mapping complex structures to simple features [Lavrac *et al.*, 1991; Kramer *et al.*, 2001; Khardon *et al.*, 1999]. These approaches attempt to decouple feature construction and model construction (learning) by devising methods to produce propositional features from structured data.

This paper is best viewed in this context, as it describes

our work on feature extraction languages for propositionalized relational learning. The study of feature extraction languages in the context of learning relational representations over structured domains needs to address at least three aspects of the problem. First, we develop and study a flexible knowledge representation for structured data, with an associated language that provides the syntax and a well defined equivalent semantics for expressing complex structured data succinctly. Second, we use this language to automate the process of feature construction by expressing 'types' of objects in the language, which are instantiated in the ground data. In particular, this process can determine the level at which learning is done (between ground literals and full relational expressions) by choosing these types appropriately. Finally, this process of re-representation of the domain should allow general purpose learning schemes, such as feature efficient linear algorithms and probabilistic representation and algorithms, to be defined over the resulting space.

The paper describes two different but complementary extraction frameworks and discusses their equivalence. The first, "functional" framework, following [Cumby and Roth, 2000], defines a set of relational formulae $\mathcal{R}$, a subset of FOL, with a functional calculus composed of so-called "Relational Generation Functions" (RGFs). These functions serve to generate elements of $\mathcal{R}$ representing (properties of) ground input data elements. The formulae could then be treated as features for a propositional learner. In this framework ground data is codified in a graphical structure on which the RGF calculus operates, and elements in the language are defined operationally via this calculus. The second, "syntactic" framework, expanding on [Cumby and Roth, 2002], provides a unified language used both in expressing structured features and in generating them. It builds on the idea of Description Logics to give a concrete syntactic form to the graphical representation of ground data introduced in the first framework. We provide a formal syntax and semantics for a specific feature description language (FDL) - but this is only one member in a family of languages, deterministic or probabilistic, that could be used within our framework. Domain elements and properties of them are "concepts" which are described, as in other description logics, in terms of *individuals* possessing attributes and roles in relation to other individuals. The equivalence of descriptions in FDL to a class of concept graphs is used to show efficient subsumption between descriptions The importance of inference with relational representations becomes clear in this paradigm. The description logic is an intermediate step and the basic inference step, subsumption, is used as a means to transform a domain element, e.g., a natural language sentence, and represent it in terms of a richer vocabulary – descriptions in our Feature Description Logic (FDL). This representation, in turn, may serve as an input to any propositional learning algorithm, including probabilistic algorithms, to yield structures in which sought after predicates are represented as functions (or conditional probabilities) over the relational descriptions.

We then discuss the extent to which the flexible operational language and the better defined syntactic language are equivalent and provide a mapping between the two. The FDL language is shown to possess a semantics similar to the subset $\mathcal{R}$ of FOL introduced earlier; and, the parameterized Feature Generating Function is shown to duplicate the operation of the RGF calculus.

Both frameworks differ from standard ILP approaches and most propositionalization techniques. Features are generated up front before any learning stage, in a data-driven way, based on background knowledge (or pre-learned knowledge) in the "type" of feature defined. This allows us to dictate the level of complexity of our intermediate representation before learning occurs, and to bypass a potentially expensive search for good features. Thus particularly expressive features that would not necessarily be generated during a search are allowed to influence our final learned function in a significant way.

Our techniques are aimed at complicated large-scale relational learning problems in which ground features, in addition to quantified predicates, play an important role in any learned classifier. This is the case in many natural language applications [Roth and Yih, 2001; Khardon *et al.*, 1999] where lexical features are an important part of the learned concept. In this cases, the potential number of features is very large and choosing a suitable learning approach in conjunction with the feature extraction approach is essential.

While the learning approach is presented here as an approach to learn a definition for single predicates, we view this in a wider context. Learning definitions may be used to enrich vocabulary describing the input data; the feature extraction technique can then be used incrementally to produce useful features again and subsequently to build up new representations in terms of those in a manner similar to the one envisioned in [Valiant, 1999]. Such a system might integrate easily into a programming platform, allowing researchers to construct large scale learning-based architectures to solve complex AI problems in areas such as natural language processing. It then becomes even more crucial that the basic components of this system are articulated in a language whose structure and meaning are well understood.

The remainder of the paper is structured as follows: Sec. 2 surveys related work. Sec. 3 explains the machine learning setting in which our frameworks can be used. Sec. 4 presents our Functional Feature Extraction Framework and Sec. 5 the Syntactic Feature Extraction Framework. The relations between the two is discussed in Sec. 6, and Sec. 7 concludes.

2 Related Work

Our work is mostly related to the work in the ILP community on the topic of learning relational concepts by propositional means. Commonly known as "propositionalization" methods, these approaches reformulate data for relational problems in terms of attribute-value feature vectors. A hypothesis is then induced over the set of these new features.

However, our language allows us to generate expressive, relational formulae – "quantified propositions" – and place this within any model construction approach. Specifically, it is possible to learn probabilistic classifiers and models over quantified propositions extracted with the our approach [Punyakanok and Roth, 2001]. Thus, it can also be viewed and compared with probabilistic approaches. Moreover, defining the "type" of features so as to dictate the abstraction level

of our intermediate representation is conceptually similar to modeling approaches such as relational probabilistic models [Friedman *et al.*, 1999], where the modeler may determine the level of abstraction and dependencies between entities, attributes and relations.

The following brief survey, however, focuses on related propositionalization approaches, specifically those that utilize a graph-based knowledge representation. See [Kramer *et al.*, 2001] for a good survey of propositionalization methods.

The most similar formulation to the approach we present is Kramer's graph-based approach for feature construction in biochemical domains. This approach [Kramer and Frank, 2000; Kramer and Raedt, 2001], uses structural features produced by a molecular feature mining program called MolFea in conjunction with SVM to learn a classifier for predicting carcinogenicity in molecules. [Kramer and Raedt, 2001], uses a version-space approach to represent a set of fragments in the input data that is more general than, and more specific than a given fragment. This is somewhat similar to our notion of defining particular "types" of features which are instantiated in the input data, however we allow the programmer to constrain the specificity of the instantiated features in a way designed to reduce overfitting.

Other graphical techniques include the method of [Geibel and Wysotzki, 1996] which, like ours, utilizes properties of proximity in a graph-based representation of the input data to restrict the range of features produced during feature construction, and [Cook and Holder, 1994; Gonzalez *et al.*, 2002], which construct features from graphical instances but restrict the number of features produced (to "good" features), blurring the line between feature and model construction.

3 The Learning Framework

The propositionalization approach presented in this work consists of a feature extraction stage – structured data elements represented as labeled graphs are converted to features representing relational and grounded properties of it – along with a general purpose propositional model generation (learning) stage that makes use of the extracted vocabulary.

In this framework, as in ILP, each observation in the domain is mapped into a collection of predicates that hold over elements in the domain. The key difference from standard ILP is that our representation of an observation may contain quantified formulae. "Examples" of this form are then given as input to a learning algorithm, that is supposed to produce a classifier to predict whether a particular target predicate holds for some particular elements. For example, we may wish to predict that for some domain elements X and Y, the predicate $father(X, Y)$ holds. To accomplish this task using standard propositional learning algorithms, we must generate *examples* in the form of lists of active propositions (features) for each predicate to be learned. Propositions of this form may either be fully ground as in the predicate $father(john, jack)$, or existentially quantified as in the predicate $\exists X\ father(john, X) \wedge father(X, harry)$. In the supervised learning setting each example will contain a label feature, which corresponds to the true relation between X and Y. An example of this sort can also serve as a negative example for other possible relations between elements that do not hold in it. Our major task then becomes to re-represent the data in a manner conducive to producing features over which we can learn a good model or a good discriminant function. This re-representation is the subject of the work described in the rest of this paper.

The feature extraction methods presented operate under the closed-world assumption, generating only the features judged to be active in the observation. All other features are judged to be inactive, or false. As it may be inefficient or impossible to list all features for a particular interpretation, this is a performance boon. Thus our learning algorithm should be able to accept examples represented as variable length vectors of only positive features. In addition, our methods provide the flexibility to generate a large number of features by designating a smaller set of "types" of features, so our learning algorithm should be able to learn well in the presence of a large number of irrelevant features.

In most of the applications of our approach we have used as the learning component, the SNoW[1] learning system. This is a multi-class propositional classifier suited to a high dimensional but sparse representation of feature data of variable length that uses a network of linear functions to learn the target concept. It has been shown to be especially useful for large scale NLP and IE problems [Khardon *et al.*, 1999; Roth and Yih, 2001; Golding and Roth, 1999]. Unlike "traditional" ILP methods that typically learn concepts represented as conjunctive rules, SNoW employs a variation of a feature-efficient learning algorithm, Winnow [Littlestone, 1988] (or other linear learning algorithms), to learn a linear function over the feature space; consequently, these "generalized rules" are more expressive than simple rules, and are easier to learn.

4 Functional Feature Extraction Framework

This section introduces a feature extraction framework in which elements in a restricted subset of first-order logic, generated using a set of composable functions with an associated calculus, serve as features for learning.

4.1 The Relational Language $\mathcal{R}$

The relational language $\mathcal{R}$ is a restricted first order language. The *alphabet* consists of (i) variables, (ii) constants, (iii) predicate symbols, (iv) quantifiers and (v) connectives. (ii) and (iii) vary from alphabet to alphabet while (i), (iv) and (v) are the same for every alphabet. Formulae in $\mathcal{R}$ are defined to be restricted function-free first order language formulae in which there is only a single predicate in the scope of each variable.

Definition 4.1 *An* atomic formula *is defined inductively:*

1. *A* term *is either a variable or a constant.*
2. *Let p be a k-ary predicate, $t_1, \ldots, t_k$ terms. Then $p(t_1, \ldots, t_k)$ is an atomic formula.*
3. *Let F be an atomic formula, x a variable. Then $(\forall x F)$ and $(\exists x F)$ are atomic formulae.*

Definition 4.2 *A* formula *is defined inductively as follows:*

[1] Available at `http://L2R.cs.uiuc.edu/~cogcomp/`

1. *An* atomic formula *is a formula.*
2. *If F and G are formulae, then so are $(\sim F), (F \wedge G), (F \vee G)$.*

The relational language given by the alphabet consists of the set of all formulae constructed from the symbols of the alphabet. We call a variable-less atomic formula a *proposition* and a quantified atomic formula, a *quantified proposition* [Khardon *et al.*, 1999]. The informal semantics of the quantifiers and connectives is as usual.

For formulae in $\mathcal{R}$, the *scope* of a quantifier is always the unique predicate that occurs with it in the atomic formula. All formulae in $\mathcal{R}$ are *closed* since all formulae are composed from propositions or quantified propositions which are connected via $\sim$, $\wedge$ or $\vee$, thus variable occurrences are *bound*.

4.2 Interpretation

$\mathcal{R}$ is used as a language for representing knowledge with respect to a domain; we now define how formulae in $\mathcal{R}$ receive their truth values.

Definition 4.3 *A domain $\mathcal{D}$ for the language $\mathcal{R}$ is a collection D of elements along with*

(i) *An assignment for each constant in $\mathcal{R}$ to an element in D.*

(ii) *For each k-ary predicate in $\mathcal{R}$, the assignment of a mapping from D^k to $\{0,1\}$ ($\{$true,false$\}$).*

When there is no confusion, we will call D, the set of elements in the domain, the domain. We can always think of D as the Herbrand base, the collection of all ground atoms in $\mathcal{R}$ [Lloyd, 1987]. In this case any interpretation is a subset of the Herbrand base, so we can talk in terms of subsets of D.

Given $D' \subseteq D$, a formula F in $\mathcal{R}$ is given a unique truth value, which we call *the value of F on D'*. This value is defined inductively using the truth values of the predicates in F, and the semantics of the connectives. Notice that if F has the form $\exists p$ ($\forall p$, resp.), for some k-ary predicate, then its truth value is *true* (1) iff there exists (for all, resp.) $d_1, \ldots d_k \in D'$ such that $p(d_1, \ldots d_k)$ has truth value *true*. Since for formulae in $\mathcal{R}$ the scope of a quantifier is always the unique predicate that occurs with it in the atomic formula, we have:

Proposition 4.4 *Let F be a formula in $\mathcal{R}$, $D' \subseteq D$, and let t_p be the time to evaluate the truth value of an atom p in F. Then, the value of F on D' can be evaluated in time $\sum_{p \in F} t_p$.*

That is, F is evaluated simply by evaluating each of its atoms (ground or quantified) separately.

4.3 Relational Generation Functions

Definition 4.5 (features) *Let D be a domain, $D' \subseteq D$. We call D' an* instance, *and $X = 2^D$ an* instance space. *A* feature[2] *is a function $\chi : X \to \{0,1\}$. χ can be viewed as an indicator function over X, defining the subset of those elements in X that are mapped to 1 by χ.*

Formulae in $\mathcal{R}$ are viewed as features over the instance space 2^D. A formula F maps $D' \subseteq D$ to its truth value on D'. A formula is *active* in D' if it has truth value *true* on D'.

[2] In earlier versions of this work features were called "relations".

Given an instance, we would like to know what are the features (with corresponding formulae) that are *active* in it. We would like to do that, though, without the need to write down explicitly all possible formulae in the domain. This is important, in particular, over infinite domains or in on-line situations where the domain elements are not known in advance, and therefore it is simply impossible to write down all possible formulae. An efficient way to do that is given by the construct of *relational generation functions*. As will be clear later, this notion will also allow us to significantly extend the language of formulae by exploiting properties of the domain.

Definition 4.6 *Let $\mathcal{X}$ be an enumerable collection of features on X. A relational generation function (RGF) is a mapping $G : X \to 2^{\mathcal{X}}$ that maps $x \in X$ to a set of all elements in $\mathcal{X}$ that satisfy $\chi(x) = 1$. If there is no $\chi \in \mathcal{X}$ for which $\chi(x) = 1$, $G(x) = \phi$.*

RGFs can be thought of as a way to define "types" of formulae, or to parameterize over formulae. Only when an instance $D' \subseteq D$ is presented, concrete formulae are generated.

4.4 Relational Calculus

The family of relational generation functions for $\mathcal{R}$ are RGFs whose outputs are formulae in $\mathcal{R}$. Those are defined inductively, just like the definition of the language $\mathcal{R}$.

The relational calculus is a calculus of symbols that allows one to inductively compose relational generation functions. The alphabet for this calculus consists of (i) basic RGFs, called *sensors* and (ii) a set of connectives. While the connectives are the same for every alphabet the *sensors* vary from domain to domain. A sensor is a way to encode basic information one can extract from an instance. It can also be used as a uniform way to incorporate external knowledge sources that aid in extracting information from an instance.

Definition 4.7 *A* sensor *is a relational generation function that maps an instance D' into a set of atomic formulae in $\mathcal{R}$. When evaluated on an instance D' a sensor s outputs all atomic formulae in its range which are* active.

Definition 4.8 *Let C be a set of formulae. A* conditioning *operation $|C$ on an RGF r restricts r to output features for only formulae in C.*

Definition 4.9 *The operation of a* relational generation function *(RGF) for $\mathcal{R}$ is defined inductively as follows:*

1. *When evaluated on an instance D' the sensor s outputs features for all* active *atomic formulae in its range.*
2. *If s and r are RGFs for $\mathcal{R}$, then so are $(\neg s_{|F}), (s \& r), (s_{|F_1} | r_{|F_2})$.*
   - **i** *The feature output by $(\neg s_{|F})$ corresponds to the formula $\neg F$ given that F is in the range of s and is not active on D'.*
   - **ii** *The features in the output of $(s \& r)$ correspond to active formulae of the form $F_1 \wedge F_2$, where F_1 is in the range of s and F_2 is in the range of r (evaluated on D').*
   - **iii** *The features in the output of $(s_{|F_1} | r_{|F_2})$ correspond to formulae of the form $F_1 \vee F_2$, where either F_1 is active in D' or F_2 is active in D'.*

Notice that for negation and disjunction it is necessary to condition the argument RGFs with input formulae, as for many sensors the range of formulae which are not active in the current instance may be infinite. For conjunction and disjunction, it is possible to focus the range of formulae to those active for a particular subset of the current instance, based on structural information as described in Sec. 4.5.

4.5 Structural Instance Space

So far we have presented $\mathcal{R}$ and RGFs with respect to an abstract domain D. In most domains more information than just a list of objects and assignments is available. We abstract this using the notion of a *structural domain* that is defined below. Instances in a structural domain are augmented with some structural information and, as a result, it is possible to define more expressive RGFs in terms of the sensors provided along with the domain.

Structured Instances

Definition 4.10 *Let D be the set of elements in the domain. A* structured instance *O is a tuple $(V, E_1, E_2, ...E_k)$ where V is a set of nodes each associated with some subset $D' \subseteq D$ of elements in the domain, and E_i is a set of edges on V. The graph $G_i = (V, E_i)$, is called the ith* structure *of instance O.*

Structural Operations

We now augment the relational calculus of Sec 4.4 by adding structural operations. These operations exploit the structural properties of the domain as expressed in the graphs G_is in order to define RGFs, and thereby generate non-atomic formulae that may have special meaning in the domain.

Definition 4.11 *Let $V' \subseteq V$ be a set of nodes in the structured instance O. An RGF r is* focused *on V' if, given an instance D' it generates features only for formulae in its range that are active on those domain elements associated with V'. The focused RGF is denoted $r[V']$.*

Definition 4.12 *Let $s_1, s_2, \ldots s_k$ be RGFs for $\mathcal{R}$. $colloc_g(s_1, s_2, \ldots s_k)$ is a restricted conjunctive operator that is evaluated on a chain of length k in the gth structure of the given structured instance. Specifically, let $O = \{G_i\}_1^m$ be a structured instance and let $v_1, v_2, \ldots v_k$ be a chain in G_i. The features generated by $colloc_i(s_1, s_2, \ldots s_k)$ are those generated by $s_1[v_1]\&s_2[v_2]\&\ldots\&s_k[v_k]$, where by $s_j[v_j]$ we mean here the RGF s_j focused to $\{v_j\} \subseteq V$, and the $\&$ operator is defined as in Definition 4.9. Notice that each subRGF conjunctions may produce more than one feature.*

Focus-Word Centered Representation

The structural information also provides an easy way to *focus* the RGFs (Def 4.11). For example, defining a set of elements for the focus set V' in $s[V']$ can be done using some graph property. Specifically, we use the notion of a focus node, and define a focus set with respect to it using a radius length. In particular, in the $colloc$ operation, we can restrict the chains to *start* at a node $v' \in V$ at a certain length in edges from a focus node v or to *contain* it. Notice that if, for the given instance $O = (V, G)$, we have that $v' \notin V$, then the output is an empty set of features.

The next section describes an alternative to the functional extraction framework based on a syntactic construction.

5 Syntactic Feature Extraction Framework

This section presents a feature extraction framework based on a description logic-like language. Statements in this language are constructed to be equivalent to a restricted graphical representation for our relational data, and they serve as input for a parameterizable *Feature Generating Function*.

5.1 Feature Description Logic

The basic Feature Description Logic (FDL) is described below by providing its formal syntax and semantics.

As in most formal description logics, FDL descriptions are defined with respect to a set X of *individuals*. However, unlike most KL-ONE-like description logics, the basic alphabet for FDL descriptions includes *attribute*, *value*, and *role* symbols. We differentiate attribute from role descriptions and our basic primitive description is an attribute-value pair. We also allow a non-functional definition of attribute descriptions and role descriptions; thus an attribute describing an individual may take many values and a role describing an individual could take several different fillers.

This type of language is useful since, at a basic level, statements in the language abstract over sets of objects in different domain instances that have that have the same attributes and relationships to other objects present. These statements therefore serve as a useful basis for features for learning algorithms. At another level, by quantifying over the set of values that attributes can take, we can describe an even more general set of individuals. The procedure which we later introduce takes statements of this type and information about the current set of objects being considered, and rewrites the statements to contain the values seen in the current instance.

Definition 5.1 *A FDL* description *over the* attribute *alphabet $Attr = \{a_1, ..., a_n\}$, the* value *alphabet $Val = v_1, ..., v_n$, and the* role *alphabet $Role = \{r_1, ..., r_n\}$ is defined inductively as follows:*

1. *For an attribute symbol a_i, a_i is a description called a* sensor. *For some value symbol v_j, $a_i(v_j)$[3] is also a description, called a* ground sensor. *We also define a special* identity sensor *denoted $*$, which represents all individuals x.*
2. *If D is a description and r_i is a role symbol, then (r_i D)[4] is a* role *description.*
3. *If $D_1, ..., D_n$ are descriptions, then (***AND** *$D_1, ..., D_n$) is a description. (The conjunction of several descriptions.)*

We also define the *size* of a description $|D|$ as the number of conjunctive and role sub-descriptions present in D. Def. 5.1 allows the recursive construction of FDL descriptions.

We now turn to the semantics of FDL descriptions. This discussion follows a model-theoretic framework similar to that laid out in [Borgida and Patel-Schneider, 1994]. This definition uses the notion of an interpretation [Lloyd, 1987], and that of an *interpretation function* which can be viewed as the function that encodes the information about domain. For a domain element z we denote by z^I its image under the interpretation function.

[3] read: a_i takes value v_j

[4] read: relation r_i holds for current object and those in $ext(D)$

Definition 5.2 (FDL extension) *An interpretation I consists of a domain Δ, for which there exists an interpretation function I. The domain is divided into disjoint sets of individuals, X, and values, V. The interpretation function assigns an element $v^I \in V$ to each value v. It assigns a set of binary relations a^I over $X \times V$ to each symbol a in Attr, and a set of binary relations r^I over $X \times X$ to each symbol r in Role. The* extension *of a FDL description $ext(D)$ is defined as follows:*

1. *The extension of a sensor is defined as $ext(a(v)) = \{x \in X | (x, v^I) \in a^I\}$. The extension of an existential sensor $ext(a)$ is $\{x \in X | \exists v^I \in V$ s.t. $(x, v^I) \in a^I\}$.*
2. *The extension of a role is defined as $ext((r\ D))$ = $\{x \in X | (x, y) \in r^I \rightarrow y \in D^I\}$.*
3. *The extension of a conjunctive expression $ext($(**AND** D_1 D_2)) is defined as $ext(D_1) \bigcap ext(D_2)$.*

We can now define the *subsumption* of a FDL description D_1 by another description D_2. We say that D_1 *subsumes* D_2 iff the extension of D_2 is a subset of the extension of D_1. i.e. $D_1^I \supseteq D_2^I$ for all interpretations I. In our framework subsumption is used to transform a domain element, represented as a concept graph, into a feature set that can serve as an input to a propositional learning algorithm.

To show that FDL allows efficient subsumption, we use the notion of a concept graph that we define next.

5.2 Concept Graphs

The notion of concept graphs stems from work in the semantic network and frame-based representations. In many ways description logics were invented to provide a concrete semantics for the construction of such graph-based knowledge representations. Here they provide a tool for computing subsumption between descriptions and as a convenient representation for examples presented to algorithms in our learning framework.

FDL concept graphs are a variation on the type invented for [Borgida and Patel-Schneider, 1994] to explain "basic CLASSIC". A FDL concept graph is a rooted labeled directed graph $G = G(N, E, v_0, l_N)$, where N is a set of nodes, $n_0 \in N$ is the root of the graph, $E \subseteq (N \times N \times Role)$ a set of labeled edges (with role symbols as labels) and l_N is a function that maps each node in N to a set of sensor descriptions.

The semantics of FDL concept graphs is defined similarly to that of basic CLASSIC, minus those associated with equality constraints. The extension of a node in the graph is intended to be the set of individuals described by its corresponding description.

Definition 5.3 (Concept Graph extension) *Given a FDL concept graph $G = (N, E, n_0, l_N)$, a node $n \in N$, and an interpretation I in some domain Δ composed of elements X and values V, we say that an individual $x \in X$ is in the extension of n iff:*

1. *For each sensor $a_i(v) \in l_N(n)$, $a_i^I(x, v^I)$ is true. For each sensor $a_i \in l_N(n)$, $\exists v^I \in V$ s.t. $a_i^I(x, v^I)$ is true.*
2. *For each edge $(n, m, r_i) \in E$, $\forall y \in X$ if $r_i^I(x, y)$ then y is in the extension of m.*

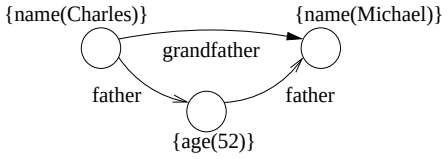


Figure 1: An example concept graph for the kinship domain.

As in earlier works on DL, an individual x is in the extension of G, iff it is in the extension of n_0. It will be clear later that in our paradigm we care about concept graph extension only as a clean way to define subsumption; the more basic notion here is the description itself. Two constructs over domain Δ are semantically equivalent if they have the same extensions given an interpretation I. The significance of concept graphs for our purposes stems from the following theorem.

Theorem 5.4 *Any FDL description D is semantically equivalent to an acyclic FDL concept-graph of size polynomial in $|D|$ that can be constructed in polynomial time.*

Thm 5.4 allows now to show that FDL supports efficient subsumption queries between descriptions and, moreover, that it supports checking subsumption of an arbitrary concept graph by a description.

Theorem 5.5 *For FDL descriptions D_1, D_2 the subsumption of D_2 by D_1 ($D_1 \supseteq D_2$) can be decided in polynomial time. Additionally for a description D_1 and an arbitrary FDL concept graph G_2, the subsumption of G_2 by D_1 can be decided in polynomial time.*

Given these definitions for FDL descriptions and their corresponding concept graph representations, it now becomes possible to describe a feature extraction framework where such representations play a major role. Efficient subsumption testing allows generation of expressive propositional features from arbitrarily complex data represented by concept graphs.

5.3 Feature Generating Functions

Up until this point, our treatment of FDL has closely mirrored that of similar CLASSIC-like DL's [Borgida and Patel-Schneider, 1994]. However, our usage of FDL descriptions is vastly different from the usage of descriptions in these other DL's. The most closely related usage may be that of P-CLASSIC [Koller *et al.*, 1997] descriptions, in which a probabilistic distribution over descriptions is used to perform probabilistic subsumption queries. Instead, in our paradigm, descriptions are used to generate propositional formulae, in a data-driven way via subsumption queries. We first describe the process of generating propositional formulae using FDL descriptions.

The essential construction of our method is a *Feature Generating Function*, closely related to the RGF of Sec. 4.3.

The constructions, however, differ in an important respect. Here we discuss a general Feature Generating Function, whose operation is constrained by the formal syntax of the generating descriptions themselves, having well defined structure and meaning. This therefore extends and unifies the "relational calculus" of Sec. 4.3 that procedurally composes different types of RGFs to produce complex features.

In fact, we claim that any operation of such a calculus may be pushed onto the syntax of an FDL, and therefore it is possible to define descriptions in our language that produce exactly the same features as produced there (as we explain later).

Definition 5.6 (features) *Let I be some interpretation with domain $\Delta = (X, V)$, and let $\mathcal{I}$ be the space of all interpretations. For a description D we define a* feature F_D *to be a function $F_D : \mathcal{I} \rightarrow \{0, 1\}$. F_D acts an indicator function over $\mathcal{I}$, denoting the interpretations for which the extension of the description D is not empty.*

Given an interpretation I we say that a feature F is *active* in I if it evaluates to true. Generating such features efficiently however is the topic of much debate, as such feature spaces could be prohibitively large or in some cases infinite, making manual generation impossible.

Our next step is to automate the construction of features of this sort. Luckily, the semantics of FDL descriptions and their equivalence to rooted concept graphs give rise to an efficient method of constructing *active* features, via the notion of the *feature generating function* (FGF). Let some interpretation I be represented as a concept graph G, in which all elements of I are in the extension of some node of G. The construction of this graph is efficient, following Thm 5.4.

Our FGF method takes G along with a set of input FDL descriptions $\mathcal{D}$, and outputs a set of active features over G. The basic method computes a *feature description* D_θ[5] for each attribute as described in Def. 5.8 for G with respect to each description $D \in \mathcal{D}$, and constructs a feature for each D_θ. The intuition is that each input description defines a "type" of feature, subsuming many possible (partially) ground descriptions over an interpretation. We say a description is *ground* if it is a description containing only sensors of the form $a_i(v_i)$.

Definition 5.7 (Feature Generating Function) *Let $\mathcal{F}$ denote an enumerable set of features over the space $\mathcal{I}$ of interpretations and let D be a description. A* feature generating function *$\mathcal{X}$ is a mapping $\mathcal{X} : \mathcal{I} \times D \rightarrow 2^{\mathcal{F}}$ that maps and interpretation I to a set of all features in $\mathcal{F}$ such that $F_D(I) = 1$.*

Thus, the image of I under $\mathcal{X}$ is a re-representation of I in terms of the (set of D_θ's subsumed by the) description D.

Definition 5.8 *The* feature description *of a rooted concept graph G with respect to an input description D is the unique ground description D_θ subsuming G and subsumed by D, containing only ground forms of the sensors in D.*

In the case that D is itself already ground, computing the feature description D_θ amounts to checking the subsumption of G by D.

As usual, the importance of features stems from the fact that they might provide some abstraction. That is, they describe some significant property of the input which may occur also in other, different, inputs.

Theorem 5.9 *Given any interpretation I represented as a concept graph and a description D, all active features over I with respect to D can be generated by $\mathcal{X}$ in polynomial time.*

[5]The θ here indicates the binding that occurs for each attribute.

6 Mapping the Two Formalisms

As previously stated, the two methods presented are both means to generate propositional features from structured or semi-structured input data. In the first case presented, a more functional approach is taken. Sensor RGFs may produce formulae inferred through some process and not explicitly provided as predicates in the domain. For example, in a visual processing problem, we might want a sensor such as the $I > 50$ sensor discussed earlier. The domain may provide predicates only of the form $intensity(60)$,

By contrast, the second method attempts to push all of the functionality of the RGF functions, and the logical structure implied by using graph operations, onto the syntax of our description language. The process of constructing features is then reduced to a single mechanical operation. In order to produce a feature from a particular domain instance, the information contained in that feature must be explicitly represented in the graphical structure, or else implied in the syntax of the language. For complicated learning problems involving several stages of classification, we thus update the domain instance with information gained from previous stages. In this manner we can, for example, simulate the learning of recursive concepts such as a $path$ in a graph. If a domain instance is given with $edge$ arcs represented between each node, we can first learn a classifier to predict that two nodes linked by an $edge$ define a $path$. After filling in all predicted $path$ arcs based on this classifier, a second classifier based on the existing $path$ arcs and the new $edge$ arcs can be learned, and iteratively applied to predict the remaining $path$ edges between any pair of nodes.

While the above discussion would give the impression that the formalism of RGFs could yield more expressivity than the FDL formalism, we claim that we can simulate most of the features output in the first framework with the second. The details of this mapping appear in [Cumby and Roth, 2003a].

Here we stress that, although the formalisms presented can be mapped to one another in terms of creating features to express the same concepts in the same situations, each has its own unique advantages. The RGF formalism serves as a more abstract foundation for feature extraction. It allows us to devise extraction functions that are independent of the underlying knowledge representation used for the data. The conjunctive $\&$ operator and the disjunctive $|$ operator exhibit this independence. The FDL based approach, by encoding particular graph properties explicitly in the syntax of the language and by directing the feature extraction process through syntactic operators, gives a more "implementation-level" understanding of that process. Additionally, with a cleanly established syntax for the description language we can substitute other functionality in place of explicit feature generation. For example, the language can be used as the basis of a parameterizable family of kernels for use with kernel learners as shown in [Cumby and Roth, 2003b].

7 Conclusion

This work presents two paradigms for efficient learning and inference with relational data. The first framework addressed feature extraction in a functional setting through the defini-

tion of Relational Generation Functions. This framework can be viewed as providing a general basis for the second framework, abstracting away many of the details of the feature construction process. For example, we introduced the abstract notion of a *sensor* RGFs, which we allowed to construct features by inferring predicates from input instances, through external functions or any other means.

The second paradigm defined the notion of feature description logics - a relational language with clear syntax and semantics that can be used, via feature generation functions, to efficiently re-represent world observations in a way that is suitable for general purpose learning algorithms. We have shown that both these formalisms allow one to efficiently learn complex relational representations, in a system in which the basic components are articulated in a language whose structure and meaning are well understood.

It is important to point out that a wide family of feature description logics can be used within our framework. In fact, other CLASSIC-like description logics, as well as their probabilistic variations, could be incorporated into the framework with the addition of a Feature Generating Function for each. They could then participate as building blocks in the process of learning relations and predicates. For example, features generated in this framework need not be defined as Boolean. They can be associated with a real number, indicating the probability the feature holds in the interpretation, allowing a immediate use of P-CLASSIC like languages. This approach provides a different view on ways to extend such description languages, orthogonal to the one suggested by existing extensions, such as PRMs [Friedman *et al.*, 1999]. Unlike those extensions, which are more suitable to relational database-like (probabilistic) inferences, we provide a natural solution to learning predicates and relational structure, as seen in the examples pointed to in [Cumby and Roth, 2003a]. Furthermore, the syntactic framework allows us to use the FDL language for tasks other than pure feature extraction. For example, in [Cumby and Roth, 2003b], a family of relational kernel functions parameterized by descriptions in the language is developed for use with the Kernel Perceptron algorithm.

Some future directions include the use of our formalism to determine in a data driven way the level of abstraction of feature 'types' required for a given application; the development of nested and hierarchical FDL-based knowledge representations; and integrating our framework into a programming platform, allowing researchers to construct large scale learning-based architectures to solve complex AI problems.

References

[Borgida and Patel-Schneider, 1994] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *JAIR*, 1:277–308, 1994.

[Cohen, 1995a] W. Cohen. PAC-learning recursive logic programs: Efficient algorithms. *JAIR*, 2:501–539, 1995.

[Cohen, 1995b] W. Cohen. PAC-learning recursive logic programs: Negative result. *JAIR*, 2:541–573, 1995.

[Cook and Holder, 1994] D.J. Cook and L.B. Holder. Substructure discovery using minimum description length and background knowledge. *JAIR*, 1:231–255, 1994.

[Cumby and Roth, 2000] C. Cumby and D. Roth. Relational representations that facilitate learning. In *Proc. of KR'2000*, 2000.

[Cumby and Roth, 2002] C. Cumby and D. Roth. Learning with feature description logics. In *Proc. of ILP'02*, 2002.

[Cumby and Roth, 2003a] C. Cumby and D. Roth. Feature extraction languages for propositionalized relational learning. Technical Report UIUCDCS-R-2003-2346, UIUC Computer Science Department, May 2003.

[Cumby and Roth, 2003b] C. Cumby and D. Roth. On kernel methods for relational learning. In *Submission*, 2003.

[Dzeroski *et al.*, 1992] S. Dzeroski, S. Muggleton, and S. Russell. PAC-learnability of determinate logic programs. In *Proc. of COLT*, pages 128–135, 1992.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *IJCAI'99*, pages 1300–1309, 1999.

[Geibel and Wysotzki, 1996] P. Geibel and F. Wysotzki. Relational learning with decision trees. In *ECAI'96*, pages 428–432, 1996.

[Golding and Roth, 1999] A. R. Golding and D. Roth. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130, 1999.

[Gonzalez *et al.*, 2002] J. Gonzalez, L.B. Holder, and D.J. Cook. Graph-based relational concept learning. In *ICML'02*, 2002.

[Kersting and Raedt, 2000] K. Kersting and L. De Raedt. Bayesian logic programs. In *Proc. of the ILP'2000 Work-in-Progress Track*, pages 138–155, 2000.

[Khardon *et al.*, 1999] R. Khardon, D. Roth, and L. G. Valiant. Relational learning for NLP using linear threshold elements. In *Proc. of IJCAI*, 1999.

[Koller *et al.*, 1997] D. Koller, A. Levy, and A. Pfeffer. P-classic: A tractable probabilistic description logic. In *Proc. of NCAI'97*, pages 360–397, 1997.

[Kramer and Frank, 2000] S. Kramer and E. Frank. Bottom-up propositionalization. In *Proc. of the ILP-2000 Work-In-Progress Track*, 2000.

[Kramer and Raedt, 2001] S. Kramer and L. De Raedt. Feature construction with version spaces for biochemical applications. In *Proc. ICML'01*, 2001.

[Kramer *et al.*, 2001] S. Kramer, N. Lavrac, and P. Flach. Propositionalization approaches to relational data mining. In *Relational Data Mining*. Springer Verlag, 2001.

[Lavrac *et al.*, 1991] N. Lavrac, S. Dzeroski, and M. Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *Proc. of the 5th European Working Session on Learning*, pages 265–281, 1991.

[Levesque and Brachman, 1985] H. Levesque and R. Brachman. A fundamental tradeoff in knowledge representation and reasoning. In R. Brachman and H. Levesque, editors, *Readings in Knowledge Representation*. Morgan Kaufman, 1985.

[Littlestone, 1988] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[Lloyd, 1987] J. W. Lloyd. *Foundations of Logic Progamming*. Springer-verlag, 1987.

[Muggleton and De Raedt, 1994] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 20:629–679, 1994.

[Punyakanok and Roth, 2001] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS'2000*, pages 995–1001, 2001.

[Roth and Yih, 2001] D. Roth and W. Yih. Relational learning via propositional algorithms: An information extraction case study. In *Proc. of IJCAI*, pages 1257–1263, 2001.

[Selman, 1990] B. Selman. *Tractable Default Reasoning*. PhD thesis, Dept. of Computer Science, Univ. of Toronto, 1990.

[Valiant, 1999] L. G. Valiant. Robust logic. In *Proceedings of the Annual ACM Symp. on the Theory of Computing*, 1999.

Individuals, relations and structures in probabilistic models

James Cussens
Department of Computer Science
University of York
Heslington, York, YO10 5DD, UK
jc@cs.york.ac.uk

Abstract

Relational data is equivalent to non-relational structured data. It is this equivalence which permits probabilistic models of relational data. Learning of probabilistic models for relational data is possible because one item of structured data is generally equivalent to many related data items. Succession and inclusion are two relations that have been well explored in the statistical literature. A description of the relevant statistical approaches is given. The representation of relational data via Bayesian nets is examined, and compared with PRMs. The paper ends with some cursory remarks on structured objects.

1 Learning from iid samples

Recall from [Cussens, 2000], the well-known correspondence between the mathematical abstractions used in statistics and the real world. This correspondence is given diagrammatically in Figure 1. This view sees Nature as a machine which probabilistically spits out data in response to questions (inputs) that we give it. In some cases (e.g. clustering, density estimation) the independent variables do not play an important role—the machine does not require any input to produce an output. This probabilistic machine has many names in the literature, it is Hacking's "chance set-up" [Hacking, 1965] and Popper's "generating conditions" [Popper, 1983].

This probabilistic machine is often taken to produce output by selecting its output from some population of possible outputs. Such a reconceptualisation is sometimes strained: "But only excessive metaphor makes outcomes of every chance set-up into samples from an hypothetical population" [Hacking, 1965, p. 25]. But it is pretty much hard-coded into the standard Kolmogorovian formalisation of probability. Kolmogorov's axiomatisation defines a probabilistic model to be a probability space $(\Omega, \mathcal{F}, P)$. Here Ω is the population, and outputs (actually subsets of Ω in $\mathcal{F}$) are chosen according to P.

In standard approaches to statistical inference (or 'learning'; the terms will be used interchangeably in this paper) we assume that the observed data is composed of *independent and identically distributed (iid)* items sampled from Ω. The homogeneity of such data permits estimation of P. To

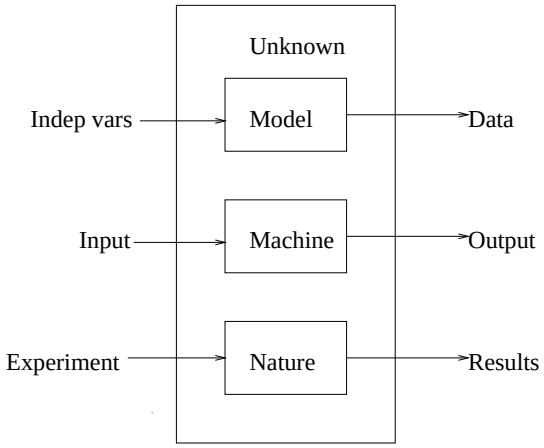


Figure 1: Statistical Inference

continue the metaphor of the machine, which will be used throughout, we assume that: each datum is generated by one run of the machine, the same machine is used for each datum, and previous outputs of the machine do not affect how it operates on future runs. The iid assumption often permits accurate estimation of parameters from data, and sometimes both parameters *and* the structure of the model/machine.

2 Relational learning

In many cases we are presented with data where the iid assumption is invalid. In such situations, let us say that we are faced with a *relational learning* problem, on the grounds that the items of data will be related in some way. What is perhaps new in current AI research in this area is that this issue is being approached in a *general* way: formalisms—often related to first-order logic—are being created where data items may be related in an arbitrary manner. However, there exists valuable work in the statistical literature which focuses on particular relationships between data. In Sections 2.1 and 2.2 we examine two specific relations: respectively, that of succession and the "isa" relation which forms the basis of hierarchical models.

2.1 Succession

Time-series analysis is a venerable form of relational learning with a large literature. It is often applied to financial data where, say, the price of pork bellies today (let's call it X_i) is not independent of its price yesterday (X_{i-1}). So, although X_i and X_{i-1} may be identically distributed they will not be independent, hence the data is not iid. Returning to the metaphor of the machine: we have the same machine each time, but the output of the last run forms part of the input of the next run.

Given that the convenient iid assumption is lacking how is learning possible? To answer this it is useful to make a quick detour into the mathematical formalism. A time-series is modelled as a *stochastic process* which is defined "as a family of random variables $\{X_i, i \in I\}$ defined on some probability space $(\Omega, \mathcal{F}, P)$" [Brockwell and Davis, 1991, p. 8]. The index set I may be discrete (as in the case of daily commodity prices) or continuous.

In learning, our goal is to estimate the underlying probabilistic model from data. Since here this model is a stochastic process it looks as if each data point must be a realisation of the stochastic process, i.e. each data point is to be a joint instantiation of all the $\{X_i, i \in I\}$. By running this sequence-generating probabilistic machine many times over we could get an iid sample each element of which is a joint instantiation. Unfortunately, in general, the machine is run just once. We will only get one such data point—since we cannot repeatedly rewind history and observe, say, the price of pork bellies on 24th Jan 1999 many times over.

But learning is still possible if we assume the joint distribution of the $\{X_i, i \in I\}$ is *structured*. Take the simplest AR(1) model (AR(1) is also known as a Markov process):

$$\forall i : X_i \sim \alpha X_{i-1} + \epsilon_i$$

where the ϵ_i are iid. Now the observation of each X_i becomes a data point and contributes towards the estimation of α. The point is that there is a *repetitive* structure. It is the same (unknown) α for all X_i. There are two further things to note here. Firstly, individuals (for example days) are not explicitly represented; only attributes of individuals, such as the price of pork bellies on that day. These attributes (the $\{X_i\}$) are directly connected without any intervening individuals of which they are the attributes. Secondly, this description requires quantification over random variables, without time-series analysts requiring a new 'first-order' probabilistic formalism.

2.2 Hierarchy

Sequence data is not the only case where iid assumptions break down. Consider the following situation (where "an assumption of exchangeability" is essentially an iid assumption, and for "covariate" read "attribute"):

> ... in studying scholastic achievement we may have information about individual students (for example, family background), class-level information (characteristics of the teacher), and also information about the school (educational policy, type of neighborhood). ... With covariates defined at multiple levels, the assumption of exchangeability of units or subjects at the lowest level breaks down, even after conditioning on covariate information. The simplest extension from a classical regression specification is to introduce as covariates a set of indicator variables for each of the higher-level units in the data—that is, for the classes in the educational example ... But this will in general dramatically increase the number of parameters in the model ... [Gelman *et al.*, 1995, p. 366]

Here it is not good enough to produce a regression model (probabilistically) mapping information specific to an individual to scholastic achievement for that individual. We also have (in ILP speak) background knowledge which is not specific to individuals. Each student is a member of a particular class, each class is contained within a particular school and each school is in a particular neighbourhood. Each of these levels in this hierarchical setup will have attributes which will affect the student's scholastic achievement.

Prefiguring a little the discussion of PRMs in Section 2.4, we can imagine a relational database system with tables for Student, Class, School and Neighbourhood. Each of these tables will have fields for information specific to objects of that class, so-called "descriptive attributes" [Getoor *et al.*, 2001]. Following [Neville *et al.*, 2003], we will call these *intrinsic* attributes. There will also be fields for "foreign keys" which contain the names of related objects from different classes. For example, Student might have fields for attendance and age as well as a foreign key field naming the class that each student is in.

The salient relationship in this case is that of inclusion or membership: each student is a member of a class, etc. Note also that students in the same class are related to each other simply by being members of the same class. Students in the same school or neighbourhood are also related, but presumably these relationships are weaker.

The option of representing all this information in a manner identical to that for individual-specific information is rejected. This rejected option corresponds to 'propositionalisation' to use ILP speak again. It will increase the number of parameters because it will increase the length of the covariate vector considerably. The same "exploding attribute-space" phenomenon tends to occur when ILP learning scenarios are propositionalised. An appropriate probabilistic model is a hierarchical one to reflect the hierarchical nature of the data:

> ... sensible estimation of these [the parameters in the model] is only possible through further modeling, in the form of a population distribution. The latter may itself take a simple exchangeable or iid form, but it may also be reasonable to consider a further regression model at this second level to allow for the effects of covariates defined at this level. In principle there is no limit to the number of levels of variation that can be handled in this way. Bayesian methods provide ready guidance on handling the estimation of unknown parameters, [Gelman *et al.*, 1995, p. 366]

To make the connection between hierarchical modelling and Bayesian computation more concrete, we will consider

a particular example using the BUGS [Spiegelhalter *et al.*, 1996] system.

2.3 Bayesian nets for hierarchical models

Consider the following `litters` probabilistic model taken from [Spiegelhalter *et al.*, 1996]. We consider survival rates in two sets of pigs. Each set of pigs contains 16 litters. "We would like to assume that the survival rates in the litters within each set are similar, but not identical." [Spiegelhalter *et al.*, 1996]. In other words, we assume that there are phenomena at the level of sets which affect survival rates. One could imagine, for example, that the two sets of litters come from two different farms. This is clearly a hierarchical set-up, but it also implies that within each set, the individual sows are related in some way.

Suppose we want to compute the probability that a piglet, born to some particular sow, will survive. If we have observed survival rates for the litters of other sows in the same set, this should effect the value of the probability we are trying to compute. How can this be done?

Here is the approach given in [Spiegelhalter *et al.*, 1996]. Let the ith ($1 \leq i \leq 16$) sow in the jth ($1 \leq j \leq 2$) set be called sow_{ij}. For each sow sow_{ij} we wish to compute p_{ij} the probability that a piglet of hers will survive. Clearly, the number of piglets born so far (n_{ij}) and the number of those that have died (r_{ij}) for sow_{ij} are pertinent intrinsic attributes.

"The simplest conjugate model is to assume the observed number of deaths r_{ij} in litter i of group [set] j is binomial with sample size n_{ij} and true rate p_{ij}, and then assume the true rates are drawn from a beta distribution with unknown parameters." [Spiegelhalter *et al.*, 1996] The crucial point is that these parameters a_j and b_j are common for all sows in set j. This probabilistic model is represented as the Bayesian net given in Figure 2. Note the use of 'plate' notation in order to compress the representation. The square box around n_{ij} indicates that n_{ij} is always assumed instantiated so no distribution need be defined for it. The corresponding BUGS language source code is given in Figure 3.

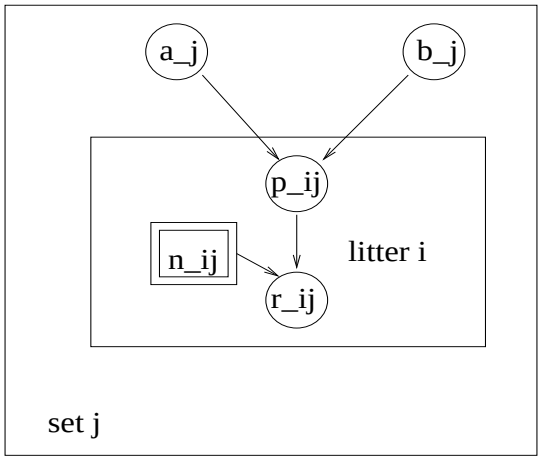


Figure 2: Bayesian net representing a hierarchical model [Spiegelhalter *et al.*, 1996]

```
for (j in 1:2) {
   for (i in 1:16) {
      r[i,j] ~ dbin(p[i,j],n[i,j]);
      p[i,j] ~ dbeta(a[j],b[j]);
   }
   a[j] ~ dgamma(1,.001);
   b[j] ~ dgamma(1,.001);
}
```

Figure 3: BUGS language representation of the Bayesian net given in Figure 2 [Spiegelhalter *et al.*, 1996]

What is interesting here is that

1. no individuals are explicity represented in the model, only attributes of individuals;
2. consequently, relationships between individuals (such as might be represented by a foreign key relationship) can not be explicitly represented, so the membership relationship between a sow and her set is not represented nor is the derived relationship between sows in a given set;
3. the individual sows in set j are related via a very abstract quantity—the parameter vector (a_j, b_j)
4. it is essential that the mediating quantity (a_j, b_j) is uninstantiated

The BUGS documentation has many other good examples of hierarchical models; the current example is one of the simpler ones.

Returning to our machine metaphor, we can say that since each litter has its own probability (p_{ij}) for piglets' surviving, there is a separate machine (specified by p_{ij}) for each litter which 'tosses a coin' and decides the fate of each piglet. However, within each set these machines are related. We model this by imagining that there is a machine-outputting machine for each set, specified by (a_i, b_i), which outputs the p_{ij} machines.

When concocting this probabilistic model it seems inconceivable that the statistician did not have particular individuals (sows, piglets) and classes (litters, sets) in mind. But by the time we have the probabilistic model all individuals have been eliminated. They merely have a ghostly presence in the indices of the random variables. It would be useful if we could find a way of formalising this elimination. To explore this question we now turn to a probabilistic formalism where individuals *are* explicity represented.

2.4 Probabilistic relational models

The ingredients of PRMs [Getoor *et al.*, 2001] are as follows. First consider *relational schemas*. A relational schema specifies a set of classes $\mathcal{X} = X_1, \ldots, X_n$. With each class X_i there is associated a set of *descriptive attributes* (i.e. attributes which individuals in that class can have) and *reference slots* these are 'attributes' whose values are the names of individuals in other classes related to individuals in this class. An *instance* of a schema defines (i) a set of individuals partitioned between the classes $\mathcal{X}$ and (ii) and values for all attributes

(real ones and instantiations of reference slots) of all individuals. A *relational skeleton* is a partial definition of an instance where only the individuals and the relations are given—the descriptive attributes are left uninstantiated. A PRM specifies a conditional probability distribution over the values of each descriptive attribute, so that given a relational schema, the PRM defines a distribution over *completions* of the skeleton. A completion of a skeleton is an instance of the schema.

PRMs are a relational 'upgrade' of Bayesian networks. Given that in Section 2.3 we have argued that at least some relational learning problems can be represented using plain old Bayesian networks, we need to examine the claimed differences between PRMs and Bayesian networks:

> However, there are two primary differences between PRMs and Bayesian networks. First, a PRM defines the dependency model at the class level, allowing it to be used for any object in the class. In a sense, the class dependency model is universally quantified and instantiated for every element in the class domain. Second, the PRM explicitly uses the relational structure of the model, in that it allows the probabilistic model of an attribute to depend also on attributes of related objects. The specific set of related objects can vary with the skeleton σ; the PRM specifies the dependency in a generic enough way that it can apply to an arbitrary structure. [Getoor *et al.*, 2001]

The first difference is inessential from a mathematical point of view, despite being important for practical model-building. We have seen that grouping together random variables associated with objects of the same class (graphically via the plate notation, or in the BUGS language using `for`) achieves the same effect. In reply one could argue that using the BUGS language is a move beyond 'plain old Bayesian nets' since it explicitly uses the quantification alluded to by [Getoor *et al.*, 2001].

The second difference is more fundamental in that a PRM holds off from giving enough information to construct an equivalent Bayesian net—the missing information is contained in the skeleton σ. The BUGS analogue is a partially specified Bayesian net, where, for example, the actual numbers of sows and piglets are yet to be determined.

Learning in PRMs assumes the data is one single structured datum:

> Our training data consists of a fully defined instance of that schema. We assume that this instance is given in the form of a relational database. [Getoor *et al.*, 2001]

So, as with time-series, the data is a single instance drawn from the underlying distribution. It is only because this instance is highly structured and hence composed of many related 'instances' that there is enough information to do parameter estimation, or possibly even model structure learning.

2.5 Eliminating and introducing individuals

It would be interesting to see whether there is an algorithm which eliminates the individuals in a PRM with a given skeleton σ thus rendering a Bayesian net (with repetitive structure) containing only "descriptive attributes". It would be even more interesting to determine the advantages and disadvantages of such a 'compilation'.

None of this is attempted here. Instead we just give one example of moving in the opposite direction. In Figure 4, we give a RDB presentation of the `litters` example. Adding the CPTs (extractable from Figure 3) for $P(\mathsf{Piglet.Lives}|\mathsf{Piglet.Mother.Health})$, $P(\mathsf{Sow.Health}|\mathsf{Sow.Set})$, $P(\mathsf{Set}.A)$ and $P(\mathsf{Set}.B)$ gives us a PRM with a given skeleton.

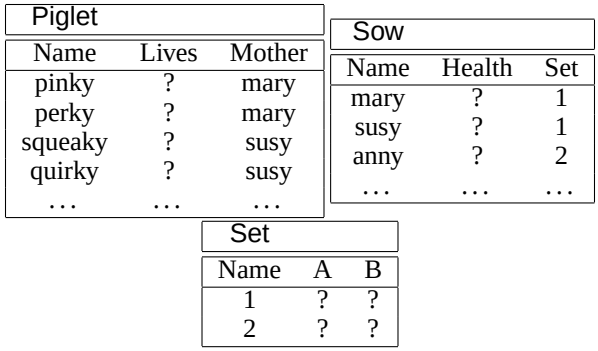


Figure 4: Relational database representation of the BUGS `litters` scenario

Comparing Figure 4 with Figures 2 and 3 the with-individuals RDB approach of PRMs seems to me to have two main advantages. Firstly, the world simply does contain individuals of various classes, and consequently this is how we conceptualise it. On this count Figure 4 is the more perspicuous to a human modeller. Secondly, and for related reasons, RDBs *are where the real-world data is*, so for entirely practical reasons a probabilistic model that can be bolted on to a RDB has a lot going for it. The advantage of the BUGS approach is that by eliminating the individuals we have gained some simplicity, or at least compactness. On a practical point the BUGS MCMC-based software is also quite well developed. All these observations indicate that 'compiling' PRMs to structured Bayesian nets may have much to recommend it.

3 Structured objects versus systems of objects

The slogan of this paper has been that relational data is equivalent to non-relational structured data. However, in the examples given the structured 'data' is a single big data-point: an entire sequence, an entire hierarchy or an entire relational database (or at least completion thereof). A less extreme kind of structured data is data composed of a number of structured objects. In place of a conclusion, in this final section we briefly consider the representation of structured objects and connections to relational data. A thorough examination of these issues and their consequences for relational learning we leave for future work.

Consider RDBs. An RDB is a system of related atomistic objects where each individual object is 'flat'. It has its own intrinsic attributes and its foreign keys name related objects.

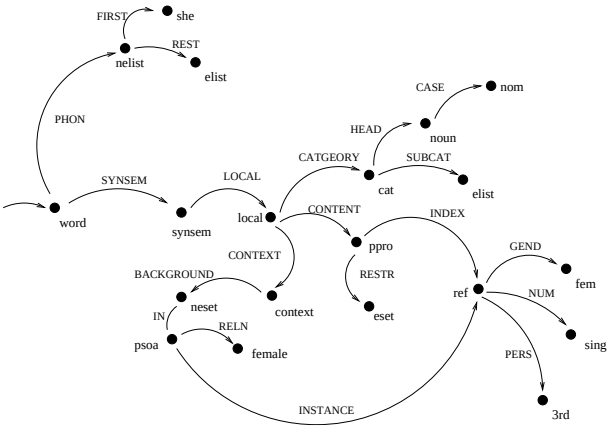


Figure 5: Lexical entry for the word "she" [Pollard and Sag, 1994, p. 17]

In logic programming terms RDBs are ground Datalog programs. However, one could argue that a putatively atomistic object (say a) with relations to other atomistic objects (say b and c) in fact has a structure such that b and c are in fact 'constituents' of a. If so, it follows that it is more perspicuous to represent a as $f(b, c)$ where f represents how b and c constitute a. Here, the information about a has been packed into a first-order term so that a mere identifier (a) has been replaced by something more like a description. The trade-offs between these two forms of representation have long been discussed in the ILP literature. For example Rouveirol [Rouveirol, 1994] shows how to 'flatten' structure representations.

An extreme example of this sort of packing occurs in lexicalised approaches to natural language grammar such as Head-Driven Phrase Structure (HPSG). In a lexicalised grammar nearly all grammatical information is represented at the word level on the grounds that words are information-rich, and should be represented as such. HPSG grammars present linguistic objects as feature-structures which are very highly structured objects. For example, Figure 5 gives a (slightly cut-down) HPSG lexical entry for the word "she" [Pollard and Sag, 1994, p. 17].

Here each node is labelled with a *sort* and the arcs correspond to features which those sorts have. Some sorts (such as *nom*) do not have features; they are called atoms. It is clear how a feature-structure could be converted to an equivalent RDB/Datalog program (in fact, this is more or less done in [Pollard and Sag, 1994]). Each arc going to a non-atom sort represents a foreign key relation; those going to atoms represent intrinsic attributes. The interesting thing here is that there is no clear distinction drawn between individuals and attributes: they are all sorts. Analogously, reference slots and descriptive attributes are all fields.

References

[Brockwell and Davis, 1991] Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods*. Springer Series in Statistics. Springer, New York, second edition, 1991.

[Cussens, 2000] James Cussens. Attribute-value and relational learning: A statistical viewpoint. In Luc De Raedt and Stefan Kramer, editors, *Proceedings of the ICML-2000 Workshop on Attribute-Value and Relational Learning: Crossing the Boundaries*, pages 35–39, 2000.

[Gelman *et al.*, 1995] Andrew Gelman, John Carlin, Hal Stern, and Donald Rubin. *Bayesian Data Analysis*. Chapman & Hall, London, 1995.

[Getoor *et al.*, 2001] Lise Getoor, Nir Friedman, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 307–335. Springer-Verlag, September 2001.

[Hacking, 1965] Ian Hacking. *The Logic of Statistical Inference*. Cambridge University Press, Cambridge, 1965.

[Neville *et al.*, 2003] Jennifer Neville, Matthew Rattigan, and David Jensen. Statistical relational learning: Four claims and a survey. In Lise Getoor and David Jensen, editors, *IJCAI-03 Workshop on Learning Statistical Models from Relational Data*, Acapulco, Mexico, August 2003.

[Pollard and Sag, 1994] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, 1994.

[Popper, 1983] Karl R. Popper. *Realism and the Aim of Science*. Hutchinson, London, 1983. Written in 1956.

[Rouveirol, 1994] Céline Rouveirol. Flattening and saturation: Two representation changes for generalization. *Machine Learning*, 14(2):219–232, 1994.

[Spiegelhalter *et al.*, 1996] D J Spiegelhalter, A Thomas, N G Best, and W R Gilks. *BUGS Examples Volume 1, Version 0.5, (version ii)*. MRC Biostatistics Unit, Cambridge, UK, 1996.

Ecosystem Analysis Using Probabilistic Relational Modeling

Bruce D'Ambrosio, Eric Altendorf, and Jane Jorgensen
CleverSet Inc..
{dambrosi, eric, jorgenj}@cleverset.com

Abstract

In this paper, we present the results of initial explorations into the application of relational model discovery methods to building comprehensive ecosystem models from data. Working with collaborators at the USGS Biological Resources Discipline and at the Environmental Protection Agency, we are engaged in two projects that apply relational probabilistic model discovery to building "community-level" models of ecosystems. A community-level ecosystem model is an integrated model of the ecosystem as a whole. The goal of our modeling effort is to aid domain scientists in gaining insight into data. Our preliminary work leads us to believe the method has tremendous promise. At the same time, we have encountered some limitations in existing methods. We briefly describe two projects and make some observations, particularly with respect to the development of synthetic, or derived, variables. We describe specific extensions we made to solve problems we encountered, and suggest elements of an extended grammar for such variables.

1. Introduction

Ecosystems are composed of interacting populations of organisms and their environments. They are notoriously difficult to study because of their size and complexity. In addition, many are unique. Controlled experimentation in these ecosystems is undesirable because of the potentially irreversible damage it may cause. However, observational data are often abundant. The challenge in studying ecosystems is to synthesize these data into coherent, comprehensive, biologically meaningful models.

While data collection traditions and techniques are mature, data analysis methodologies are less well developed. Generally, individual, domain-specific teams (e.g., a team of physicists or a team of biologists) apply traditional statistical methods to investigate pair-wise correlations among variables in their separate datasets, but have no methods for investigating the complex, noisy, cross-disciplinary interactions that are crucial to understanding the ecosystem as a whole. As a result, the standard ecosystem-level computational scientific method is a form of "generate and test": the manual construction of mechanistic models and model selection by comparing simulation results to data or expert knowledge. Probabilistic models of ecosystems are slowly becoming more common, however these have been constructed using knowledge-engineering (Kuikka *et al.,* 1999, Marcot *et al.,* 2001).

Most of the data collected in studies of ecological systems is stored in relational databases. An emerging family of methods for relational learning [Muggleton and De Raedt, 1994], [Van Laer and De Raedt, 2001], [Quinlan, 1996], [Getoor *et al*., 1999] provide the opportunity to learn comprehensive models directly from these relational data sources.

In this paper, we present the results of initial explorations into the application of model discovery methods to build comprehensive ecosystem models from data. Working with collaborators in the USGS Biological Resources Discipline and the Environmental Protection Agency, we are engaged in two projects that apply probabilistic relational model discovery to build "community-level" models of ecosystems. (A community-level ecosystem model is an integrated model of the ecosystem as a whole.) The goal of our modeling effort is to aid domain scientists in gaining insight into data and to construct complex prior hypotheses about the ecosystems studied. Our preliminary work leads us to believe the method has tremendous promise. At the same time, we have encountered some limitations in existing methods. We briefly describe two projects and make some observations, particularly with respect to the development of "synthetic", or derived, variables.

Probabilistic relational model discovery methods exploit a relational data model to derive parameters that account for variation in the explicit variables in a data model. In a Hollywood database, for example, an *actor's* income may be related to the *number* of *movies* in which s/he played a *role*. [Getoor *et al*., 1999] introduce the concepts of a *path* (a chain of references – e.g. "actor.role" above), and a terminal *aggregator* (e.g., "number" or count above) as defining a space of synthetic variables. We have found this framework useful, but limited in its ability to account for all known interactions in our data. We will describe examples motivating the introduction of two additional features, *selectors* and *variables*, into a synthetic variable grammar.

2. Applications

CleverSet is currently engaged in two ecological modeling projects: community-level modeling of the Crater Lake ecosystem (USGS) (Jorgensen *et al.,* 2003) and community-level modeling of West Nile virus disease transmission (Orme-Zavaleta *et al.,* 2003).

Crater Lake

Data

The National Park Service is concerned about long-term changes in the clarity of Crater Lake, a national park and the clearest deep-water lake in the world. Although many domain-specific surveys have been undertaken, the analytical framework necessary to link these analyses into one overall assessment of lake health has been lacking. Our goal in this project has been to formulate multiple, complex, simultaneous hypotheses given all the data obtained from the long-term studies of the lake (Larson *et al.*, 1993). These data have been collected using varying time and spatial scales. For example, surface weather condition information is available on a daily basis, but phytoplankton densities are measured only once or twice a month (and not at all in winter), while rocket-borne instrumentation to gather weather data at altitude is only rarely available.

Method

In an initial Crater Lake analysis performed for USGS, we chose a set of temporal units to frame the analysis. These units were time periods corresponding to observed patterns of clarity of the lake and for which data were available: June-July, August, September-October. We then added a table containing these time units (this unary relation establishes the basic time scale), and relating hydrological seasons annually (this binary relation establishes the basic unit of time-lag to be considered in the analysis), and related the data tables we wished to include in the analysis to this temporal table. A complete schema for the analysis is shown in Figure 1.

Results

Figure 2 shows the essential elements of the discovered model (we omit some schema elements for clarity). One relationship we discovered is that the dominant fish species in gill net catches was probabilistically dependent upon Secchi descending depth (water clarity) in the current year, mean fish weight in the current year, descending Secchi depth the previous year and dominant fish species two years previous. This and findings concerning age class structure agreed with the anecdotal evidence that schools of Kokanee smolts swimming at the edges of the lake were preyed upon by mature Rainbow trout, where they were caught in gill nets. This phenomenon does not occur every year. A time lag of two years, discovered by the model, is consistent with experts' observations. The relation between this interaction and water quality was previously unknown. Other somewhat surprising discoveries include: (1) the centrality of water clarity (measured by the Secchi "DesDepth" parameter); and (2) the lack of a direct relationship between Zooplankton count and water clarity, at least at the spatio-temporal scale studied. These finding suggest that fish attributes may serve as a predictor of water clarity.

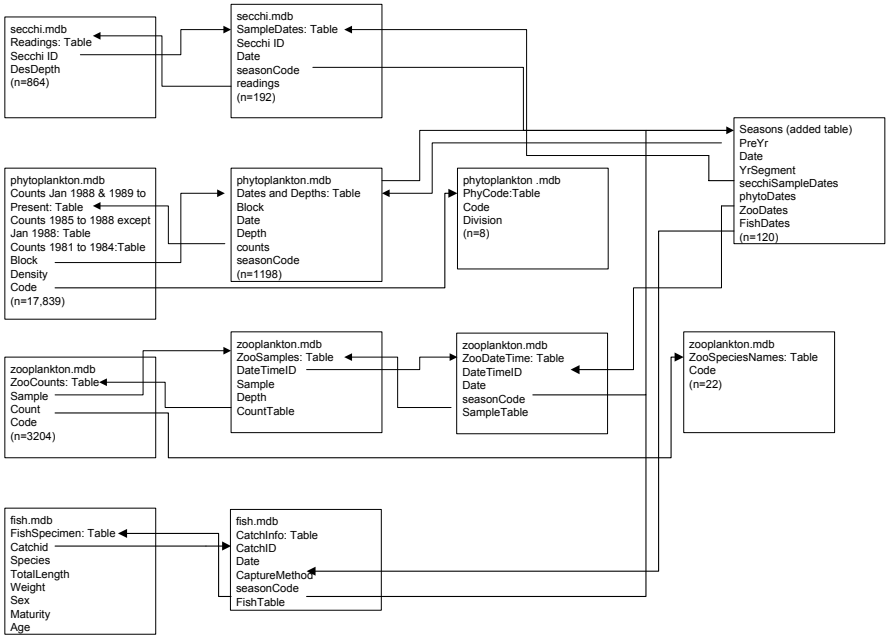


Figure 1. Crater Lake Schema

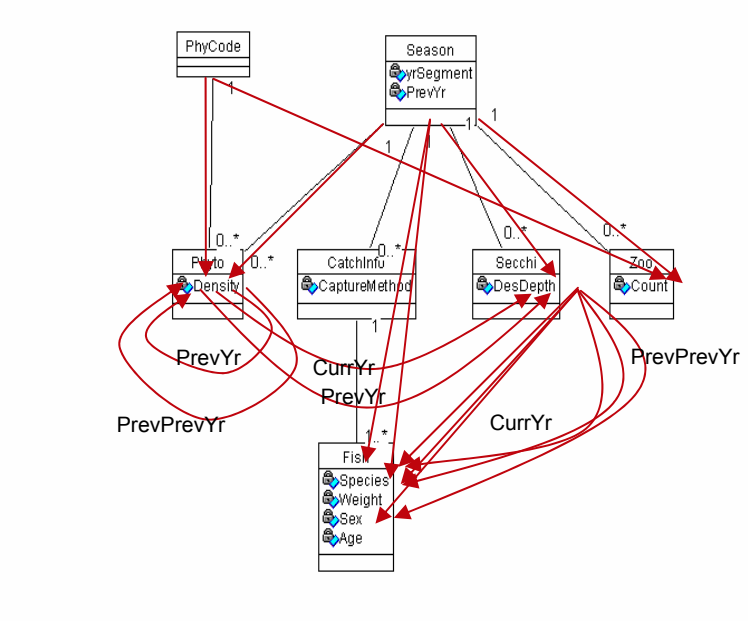


Figure 2. Crater Lake PRM

Discussion

The Crater Lake project highlighted the centrality of time in such analyses. Time creates several challenges for relational model discovery:

1. Time is rarely reified in relational schema. This presents a problem in constructing paths like "secchi.DesDepth.yrSegment.Phyto.density." Our solution in this case was to manually add a "Season" table. We have since implemented facilities for partially automating this process, by recognizing and re-ifying data/time information in schema's.
2. Once time was reified, two further decisions were necessary: we established an aggregation unit for time and we separately established a lag duration. Expert knowledge was used to establish both, based on domain knowledge and understanding of the goals of the modeling. In future we hope to explore extensions of existing statistical time series analysis methods to aid in this process.

A second problem that arose in this analysis was the frequent desire to form synthetic variables outside the scope of the current path language. For example, there were times when prior knowledge suggested that the density of a particular phytoplankton species might be a relevant parameter. Our current synthetic variable grammar does not allow for selection of a subset of the items retrieved by a path.

Finally, the goal of this project was to gain scientific insight into data that had been collected over 25 or more years (Secchi depth readings go back to the 1880s!). We found that learning models over not just the variables in the provided tables, but over their parents as well, provided additional insight. An example fragment from such an extended model, for the FishSpecimen table and its immediate parents, is shown in Figure 3. This extended model shows interactions not obvious in Figure 2, such as the multiple pathways through which Mean Secchi depth (two years previous) interacts with current Mean fish age.

West Nile Virus

Data

While the Crater Lake project involves building a relational model over multiple databases of similar type, our work with the EPA on modeling the spread of West Nile Virus involves combining multiple databases of differing types. One class of database contains incident reports (e.g., reports of dead birds testing positive for WNV, report of pools of water in which breeding mosquito populations test positive for WNV, human case reports, etc.). Each database contains reports of one type of event, located in place and time. A second class of database contains records of static features, such as the presence of a tire disposal facility (potential mosquito breeding site) or landscape type at a location. The challenge was to integrate these multiple databases into an overall model of West Nile Virus spread.

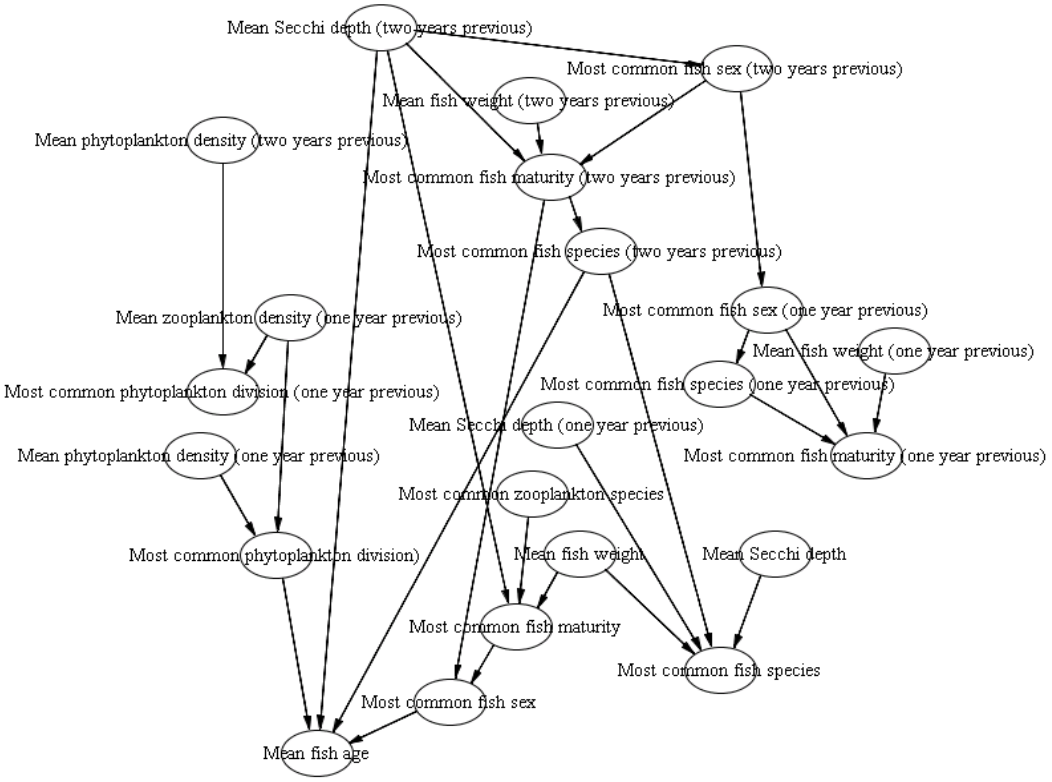


Figure 3. FishSpecimen Unrolled Model

Method

The first step in our integration of these data sources was the construction of an integrated data schema across these data sources through the addition of intensional relations linking the information in space and time. Knowing that each database recorded location in columns labeled *latitude* and *longitude*, and time as *day/month/year*, enabled us to construct a common spatio-temporal frame of reference. The simple recognition of point location in space and time, however, is not enough to integrate these data sources. Rarely do two events occur at precisely the same place or time. Rather, we imposed a *scale* across both the spatial and temporal dimensions. The parameters of this scale (five miles for space, and one month for time) were drawn from scientific knowledge about the life cycle of the vector of interest, the mosquito, and the typical flight distance for the competent bird host. Again, this was done by hand in our preliminary studies to date.

Results

Figure 4 shows a preliminary model of the spread of West Nile Virus in Maryland in 2001. Shown is a model over the synthetic variables constructed starting from the table of positive bird records.

The results support previous hypotheses that tire disposal site license density is correlated with incidence of West Nile Virus in birds. Tire disposal facilities may affect disease spread directly, by serving as breeding areas for mosquitoes, or may be a proxy for population density, which may in turn affect sampling and/or disease prevalence (e.g., though human movement through the region). The results also suggest that disease prevalence in mosquito pools may be a predictor of disease appearance in birds. The number of human and horse cases in 2001 was too small to support any significant findings related to these cases. However, even with these sparse data, the model produced is consistent with current knowledge regarding the manner in which the disease is transmitted and forms a framework in which future findings may be evaluated. The fact that horse cases do not contribute significant information to the model provides preliminary evidence that monitoring this incompetent host may be unnecessary in tracking the spread of this disease.

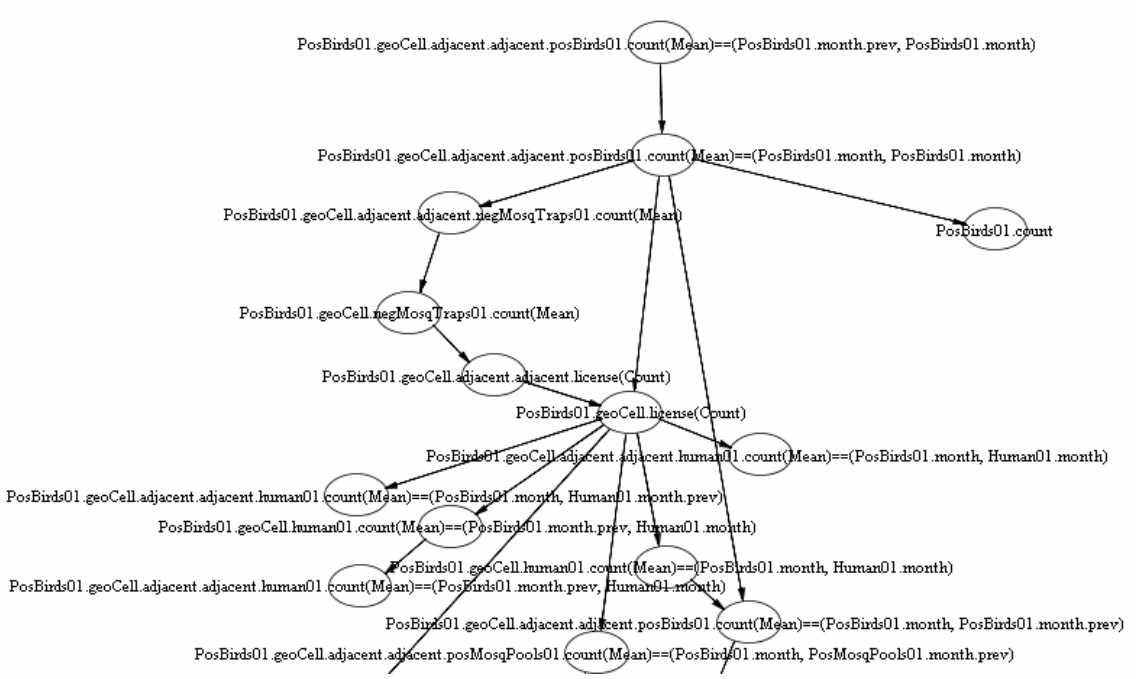


Figure 4. West Nile Virus Model Fragment

Since the mechanistic model of disease spread is not completely known, the temporal and spatial models included in the model may not be the only, or even the most useful scales at which to view interactions. Finer spatial resolutions, for example, might provide evidence about the species of birds and mosquitoes involved in transmission. Landscape level data, for example, landcover type, might also improve the descriptive and predictive capabilities of the model. As mentioned in our discussion of the Crater Lake study, our current manual methods do not permit easy exploration of possible scales.

Discussion

Our work on West Nile Virus propagation reinforces the need for selectors in synthetic variables. Unlike Crater Lake, however, where the selectors where over the values of primitive attributes, in the analysis of West Nile Virus, we needed to form equality selectors over entities (e.g., positive mosquitoes in adjacent geocells *in the same month*). We extended our synthetic variable grammar to include a single *selector* phrase. A selector is a Boolean operator mapped over the elements of the base path defining a synthetic variable. Elements for which the selector returns *true* and included in the result, and elements for which it returns *false* are omitted. The selector consists of a Boolean operator and two paths. The first path is applied to the table entry at the head of the base path for the synthetic variable, and the second path is applied to each table entry retrieved by the base path. For example, consider:

PosBirds.GeoCell.PosMosq ==(PosBird.month, PosMosq.month).Count()

The base path ("PosBirds.GeoCell.PosMosq") yields a set of positive mosquito entries in the same spatial region as a bird entry. The selector ("==(PosBird.month, PosMosq.month)") then filters out all entries not in the same month as the positive bird record. Finally, the "Count()" aggregator returns a scalar, the cardinality of the resulting set[1].

3. Conclusions and Future Work

Relational probabilistic modeling provides a natural framework for investigating ecological data. The large amount of observational, noisy data, often collected by multiple investigators over varying time-scales, provides a rich field for probabilistic model discovery, and relational approaches raise the level of modeling to one with which domain scientists can readily interact.

Existing synthetic variable construction methods naturally generate many variables either previously

[1] In more recent work, supported by NSF SBIR DMI-0231961, we have developed a more comprehensive synthetic variable language grammar and automated generation capability, patent-pending.

known to scientists or immediately recognized by them as scientifically relevant. At the same time, attempts to apply relational probabilistic model discovery techniques to ecological data have revealed limitations in our current synthetic variable construction methods. We are currently exploring work in data base *path expressions*, for example that of Van den Bussche [Van den Bussche *et al.*, 93] and Frohn [Frohn *et al.*, 94], as generalizations capable of expressing a more comprehensive set of synthetic variables. Key concepts include the *selector* and the introduction of *variables* (to allow subsequent reference to earlier elements in a path). We are also exploring mixed-initiative search procedures over these much larger path grammars.

Acknowledgements

The authors wish to acknowledge financial support from the USGS for some of the work reported here. We also thank Gary L. Larson, Research Manager, USGS Forest and Rangeland Ecosystem Science Center, Corvallis, OR and Jennifer Orme-Zavaleta, Associate Director for Science, USEPA/
NHEERL/WED, Corvallis, OR for their involvement and support.

References

[Frohn *et al*., 1994] Frohn J., Lausen G., Uphoff H., "Access to objects by path expressions and rules", Proceedings of 20th International Conference on Very Large Databases, 1994.
http://citeseer.nj.nec.com/frohn94acces.html

[Getoor *et al*., 1999] Getoor, L., N. Friedman, D. Koller, and A. Pfeffer. 1999. Learning probabilistic relational models. In *Proceedings of Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-1999),* Stockholm, Sweden. http://citeseer.nj.nec.com/context/889848/71217

[Jorgensen *et al.,* 2003] Jorgensen, J., B. D'Ambrosio, and P. A. Rossignol. 2003. Data-Driven Construction of Community Models of Crater Lake. *NSF Biocomplexity Workshop -The vertical organization of energy, carbon, and nutrient cycles in an ultraoligotrophic ecosystem: A workshop on Crater Lake, Oregon*. February 16 – 18, 2003.

[Kuikka *et al*., 1999] Kuikka, S , M. Hilden, H. Gislason, S. Hanson, H. Sparholt, and O. Varis. 1999. Modeling environmentally driven uncertainties in Baltic cod (*Gadus morhua*): Management by Bayesian influence diagrams. *Canadian Journal of Fisheries and Aquatic Sciences*. 54:629-641.

[Larson *et al*., 1993] Larson, G. L., C. D. McIntire and R.W. Jacobs, editors. 1993. Crater Lake Limnological Studies Final Report. National Park Service, Seattle, WA. 722 pp.

[Marcot *et al*., 2001] Marcot, B. G., R. S. Holthausen, M. G. Raphael, M. M. Rowland, and M. J. Wisdom. 2001. Using Bayesian belief networks to evaluate fish and wildlife population viability under land management alternatives from an environmental impact statement. *Forest Ecology and Management* 153:29-42.

[Muggleton and De Raedt, 1994] Muggleton, S. and L. De Raedt. 1994. Inductive logic programming: Theory and methods. Journal of Logic Programming, 20: 629-679.

[Orme-Zavaleta *et al.,* 2003] Orme-Zavaleta, J., J. Jorgensen, B. D'Ambrosio, H-K. Luh and P. A. Rossignol. 2003. Data-driven discovery of temporal and geospatial patterns of disease transmission: West Nile Virus in Mayland. *National Conference on USGS Health-Related Research – Natural Science and Public Health: Prescription for a better Environment*. April, 2003.

[Quinlan, 1996] Quinlan, J. R. 1996. Learning first-order definitions of functions. *Journal of Artificial Intelligence Research,* (October) 5:139-161.

[Van den Bussche and Gottfried, 1993] Jan Van den Bussche and Gottfried Vossen. An extension of path expressions to simplify navigation in object-oriented queries. In Stefano Ceri, Katsumi Tanaka, and Shalom Tsur, editors, Deductive and ObjectOriented Databases, pages 267--282, Phoenix, Arizona, 1993. Springer Verlag, Lecture Notes in CS, No. 760. http://citeseer.nj.nec.com/vandenbussche93extension.html

[Van Laer and De Raedt, 2001] Van Laer, W. and L. De Raedt. 2001. How to upgrade propositional learners to first order logic: Case study. *Lecture Notes in Computer Science*, 2049:102.

Research on Statistical Relational Learning at the University of Washington

Pedro Domingos, Yeuhi Abe, Corin Anderson,[1] AnHai Doan,[2] Dieter Fox, Alon Halevy, Geoff Hulten, Henry Kautz, Tessa Lau,[3] Lin Liao, Jayant Madhavan, Mausam, Donald J. Patterson, Matthew Richardson, Sumit Sanghai, Daniel Weld, Steve Wolfman
Department of Computer Science and Engineeering
University of Washington, Seattle, WA 98195-2350
pedrod@cs.washington.edu

Abstract

This paper presents an overview of the research on learning statistical models from relational data being carried out at the University of Washington. Our work falls into five main directions: learning models of social networks; learning models of sequential relational processes; scaling up statistical relational learning to massive data sources; learning for knowledge integration; and learning programs in procedural languages. We describe some of the common themes and research issues arising from this work.

1 Introduction

The machine learning group at the University of Washington is pursuing applications in viral marketing, Web search, adaptive Web navigation, assisted cognition, planning, knowledge integration, and programming by demonstration. In each of these areas, we began with methods that were either statistical but not relational or vice-versa, but the need for statistical relational learning (SRL) rapidly became apparent. As a result, our current focus is both on fundamental issues in SRL that cut across these applications, and on propagating advances in the fundamental issues to the applications. What follows is an overview of these research directions, showing how the need for SRL arose in each application, what fundamental issues we uncovered, what progress we have made, and the wealth of problems that remain for future work.

2 Social Networks

Statistical models of customer behavior are widely used in direct marketing. Typically, these models predict how likely the customer is to buy a product based on properties of the customer and/or the product. We have extended these models by also taking into account the *network of influence* among customers [Domingos and Richardson, 2001; Richardson and Domingos, 2002b]. This takes "word of mouth" effects into account—the fact that a customer's decision to buy is affected by what her/his friends and acquaintances say about the product. This makes it possible to design optimal *viral marketing* strategies, which choose which customers to market to based not only on their likelihood of buying, but also on their likelihood of influencing others to buy, and so on recursively. We mine these models from online sources like collaborative filtering systems and knowledge-sharing sites. We have found experimentally that they can lead to much higher profits than traditional direct marketing.

We have also worked on extending Google's PageRank algorithm for Web search with information about the content of pages [Richardson and Domingos, 2002a]. Instead of a universal PageRank measure, we introduce a query-dependent PageRank, and show how to efficiently pre-compute the necessary information at crawl time. Although superficially very different from the viral marketing problem, this problem is in fact isomorphic to it, with the words on Web pages corresponding to customer attributes, and the links between pages corresponding to social relations among customers. (See also [Chakrabarti *et al.*, 1998].)

Notice that, if we view each customer or Web page as a sample, as is usually done, these models imply that samples are no longer independent. Dependence between samples is perhaps the single most fundamental issue that arises in SRL. Even if a domain contains multiple classes of objects, each with different attributes, if the objects are all independent the joint distribution of their attributes decomposes cleanly into a product of distributions for the individual objects. This is the usual non-relational case, with the sole difference that the probabilities for all objects are not all of the same form. It is particularly remarkable that the space of models that assume sample independence is a minuscule fraction of the space of all possible models. In a sense, once the sample independence assumption is made, all further assumptions made by learning algorithms (e.g., choice of representation) are second-order perturbations.

Early studies of the issue of sample dependence in SRL include [Jensen and Neville, 2002b; 2002a], but the area is still very much in its infancy. We are currently developing general methods for this problem, based on assuming inter-sample dependences that are arbitrary but limited in number (the same type of assumption that Bayesian networks make for inter-variable dependences within a sample).

[1] Current affiliation: Google, Inc.
[2] Current affiliation: University of Illinois at Urbana-Champaign.
[3] Current affiliation: IBM T. J. Watson Research Center.

3 Relational Stochastic Processes

Large Web sites are hard to navigate—finding the information the user is looking for often takes too long, and the user gives up and/or wastes time. A possible way to ameliorate this is to automatically adapt the Web site to the user, by predicting what s/he is looking for [Perkowitz and Etzioni, 1997]. For example, we can add to the current page shortcuts to the five pages the user is most likely to want to see. We initially did this using a simple Markov model with pages as states and links as transitions, but found that, although successful, this approach had significant limitations [Anderson *et al.*, 2001]. Predictions can only be made for pages that the user has visited before (and reliable predictions only for pages that the user has visited multiple times). On large Web sites, this is a vanishingly small fraction of all the pages available. Further, as Web sites change over time, it is not possible to make predictions for new pages when they appear. Finally, generalization across Web sites is not possible: even if the adaptive Web navigation system knows the user often goes from the "Books" page to the "Science Fiction" page at Amazon.com, it cannot infer that s/he is likely to do the same at BarnesAndNoble.com.

To overcome these problems, we introduced *relational Markov models (RMMs)* [Anderson *et al.*, 2002]. RMMs model each page as a tuple in a relation, rather than an atomic state. Different pages can belong to different relations (e.g., pages about books will have different properties from pages about consumer electronics products). The variables in each relation can have hierarchically structured domains (e.g., a hierarchy of categories and subcategories of products). We consider all the abstractions of a page that can be obtained by climbing these hierarchies, and compute transition probabilities for the most informative abstractions. These probabilities are then combined into a "ground-level" prediction using shrinkage [McCallum *et al.*, 1998]. Useful predictions can thus be made for previously unvisited pages, by shrinking to abstractions of them that have been visited before (e.g., "Science Fiction Books").

RMMs are an example of a statistical relational model for a sequential domain. (See also [Friedman *et al.*, 1998; Kersting *et al.*, 2003].) However, they are still a restricted representation, in the same way that hidden Markov models are a restricted form of dynamic Bayesian network (DBNs) [Smyth *et al.*, 1997]. We are currently working on a natural generalization: dynamic probabilistic relational models (DPRMs), which extend PRMs [Friedman *et al.*, 1999] to sequential domains in the same way that DBNs extend Bayesian networks. Most processes in the world involve multiple objects and relations and evolution over time, and DPRMs should therefore be widely applicable. For example, in the viral marketing domain, we can model the spread of a product from customer to customer over time, and optimize our marketing actions at each time step, instead of our initial "one-shot" approach.

A key issue in DPRMs, as in DBNs, is efficient inference. The vastness of relational spaces, where the value of a relational variable can be any object in a given class, makes it particularly thorny. We have extended the particle filtering inference method [Doucet *et al.*, 2001] to the relational domain by Rao-Blackwellising [Murphy and Russell, 2001] relational variables conditioned on propositional ones. Initial results show that this approach is extremely effective [Sanghai *et al.*, 2003]. We are currently working on relaxing the assumptions it requires.

DPRMs are well suited to the problem of probabilistic plan recognition — that is, the task of inferring a person's cognitive state in terms of plans and intentions. The Assisted Cognition Project [Kautz *et al.*, 2003] is using DPRMs to track the behavior of a person suffering from cognitive limitations (such as mild dementia) as they go about their day-to-day activities, in order to provide pro-active help in cases of confusion and cognitive errors. Part of this work involves developing techniques for efficiently encoding hierarchical plan networks.

4 Relational Markov Decision Processes

Factored Markov decision processes (MDPs) have proven extremely successful for solving planning tasks in the presence of uncertainty, but they share the same representational weakness which we discussed in the context of Markov models and DBNs earlier. It is natural, therefore, to extend DPRMs to create relational MDPs (RMDPs). Here, state variables are relational fluents instantiated over a set of domain objects, actions are likewise parameterized, and a reward function specifies how much utility is derived from each action and its outcome. The task is to create a control strategy (called a policy) which will maximize the agent's expected discounted reward.

While it is theoretically possible to expand an RMDP into a traditional (ground) MDP, the resulting MDP is often so large that existing value and policy iteration algorithms are incapable of finding a policy. Previous researchers have proposed symbolic methods for decision-theoretic regression [Boutilier *et al.*, 2001], but these techniques are impractical. Instead, we propose generating first-order policies for RMDPs in a three step process [Mausam and Weld, 2003]. First, we create a number of ground MDPs, by instantiating the RMDP with a small set of representative objects. Second, we solve these traditional MDPs with value or policy iteration. Third, we use first-order regression to generate the high-level policy. Our approach is similar to that of Yoon *et al.* [Yoon *et al.*, 2002], but we consider a much more expressive policy representation.

5 Scaling Up

The "killer apps" of SRL are likely to be in domains where the sources of data are vast and varied. In small domains, propositionalizing the problem at some cost in human labor is often feasible. However, given that the space and time cost of a join are worst-case exponential in the number of relations being joined, in large domains this will generally not be an option. Many relational learners work by propositionalizing parts of the data on the fly (e.g., by adding attributes of related objects to the attributes of the objects of interest), and applying a propositional learner to the result [Dzeroski, 1996]. Doing this efficiently is a key but difficult problem, particularly when the relations involved do not all fit in main memory, and

must be read from disk. We are currently addressing this using subsampling techniques in two ways [Hulten *et al.*, 2003]. The first is to minimize the number of tuples that need to be read and joined, while ensuring that the sufficient statistics (and consequently the model) obtained from them is essentially the same that would be obtained from the full database. The second is to minimize the number of tuples that are used in computing an aggregate (e.g., sum, average, count), again ensuring that the result is not significantly different from what we would obtain using all the relevant tuples. This is based on our previous work in applying subsampling techniques to propositional learners [Domingos and Hulten, 2000; Hulten and Domingos, 2002]. Beyond this, we envisage that intelligent control of which tuples a learner looks at, and which join paths it pursues, will be key to scalable SRL. Heuristics for this are thus an important area of research.

6 Knowledge Integration

In traditional learning, data must first be gathered, cleaned, integrated and massaged into a single table. This process typically consumes the majority of the resources of a machine learning project. A key part of the promise of SRL is its potential to reduce or bypass parts of it: a statistical relational learner could in principle gather its own data across multiple sources, including different databases, the Web, etc., as needed for learning. However, to fulfill this potential, SRL must be able to bridge the differences in vocabulary that disparate data sources inevitably exhibit: different ontologies, different names for the same attributes, different representations of the same object, etc. Fortunately, SRL techniques can themselves be applied to help solve this "Babel problem." Given some manually created mappings between information sources, we can learn generalizations of them that allow us to map new sources automatically. We have done this successfully for relational and XML data [Doan *et al.*, 2001; 2003b] and for Semantic Web ontologies [Doan *et al.*, 2002] for the case of one-to-one mappings, and are currently extending our approach to many-to-one mappings [Doan *et al.*, 2003a]. This approach is based on using a variety of learners to extract different kinds of mapping knowledge, combining their outputs with a meta-learner, and combining the result with different types of constraint, domain knowledge, and user feedback to produce the final mapping.

More generally, SRL lends itself particularly well to knowledge-intensive learning, because it allows input knowledge to be expressed in a rich relational language, and is potentially tolerant of noise in this input. We have designed an architecture for incorporating knowledge from a large number of sources into a learner, which uses SRL techniques to handle inconsistency among sources and high variability in source quality [Richardson and Domingos, 2003a]. Specifically, we use a Bayesian logic program representation [Kersting, 2000], with knowledge-based model construction to extract the Bayesian network required to answer a given query [Ngo and Haddawy, 1997]. Horn clauses with the same consequent are combined using a noisy OR, logistic regression or logarithmic pool. The coefficient of a clause in this combination is effectively the system's estimate of the quality of the clause, and is estimated from query answers and evidence using the EM algorithm [Koller and Pfeffer, 1997]. We have successfully applied this approach in a printer troubleshooting domain. We are also exploring the use of social network models to form estimates of the quality of knowledge contributed by different users, bootstraping each user's assessment of the quality of a few others to the entire network of contributors [Richardson *et al.*, 2003].

In general, many different types of knowledge can potentially be integrated into SRL, and we are exploring this spectrum. One such type of knowledge is statements about the dependencies among variables of interest (i.e., about the structure of the Bayesian network representing the joint distribution of these variables). We have developed a method for combining statements from a variety of noisy, inconsistent sources into a single probability distribution over the network structure [Richardson and Domingos, 2003b]. This distribution can then be used as the structure prior in a standard Bayesian network learner. The method is based on postulating a simple generative model for expert statements given the true network, and inverting this using Bayes' theorem to obtain a distribution over possible networks. Our experiments show that even a small number of noisy sources can be sufficient to obtain high-quality estimates of the structure, and high-performing models as a result. We are currently extending this approach to allow Horn rules as an additional form of noisy, partial knowledge about an underlying probability distribution. Based on our experience in the printer troubleshooting domain, we expect this to be more flexible and effective than the more traditional form of knowledge-based model construction.

7 Learning Procedures

We believe that the goal of SRL should be to learn statistical models of any type of structured information, not just (for example) relational databases or Horn knowledge bases. This includes statistical models of procedures performed by humans, and of programs in procedural languages (e.g., Java, Python, C/C++). We have been pursuing applications in programming by demonstration (PBD), where the learner infers a general procedure from examples of its execution by a user (e.g., changing bibilography from one format into another). We initially approached this in a non-statistical setting, defining version spaces over procedures, and defining a version space algebra to build up complex version spaces from "atomic" ones via operations like union and join [Lau *et al.*, 2003b]. We applied this in the SMARTedit system, which learns text-editing procedures by demonstration. Our experience with this system led us to extend the version space algebra with probability distributions over version spaces, to allow incorporating knowledge from the PBD application designer on which (sub)procedures are more and less likely, and to be more flexible and noise-resistant in recognizing procedures. This can be crucial in arriving at a "best guess" as to what the user's intentions are in any given interaction. More recently, we have begun to extend this framework to learning programs with a full range of programming constructs [Lau *et al.*, 2003a].

8 Conclusion

This paper presented an overview of recent research on statistical relational learning at the University of Washington. Our work spans applications, fundamental issues, and the interplay between them. Applications we are working on include Web search, Web personalization, viral marketing, assisted cognition, planning, information integration, and programming by demonstration. Fundamental issues we have begun to make progress on include: learning in the presence of interdependencies among samples; modeling stochastic dynamics in relational domains; scaling up; learning across sources with different representations; and extending SRL beyond Horn clauses and relational databases.

Acknowledgments

This research was partly funded by an NSF CAREER Award and an IBM Faculty Partnership Award to the first author, ONR grants N00014-02-1-0408 and N00014-02-1-0932, NASA grant NAG 2-1538, and a gift from the Ford Motor Co.

References

[Anderson *et al.*, 2001] C. Anderson, P. Domingos, and D. Weld. Adaptive Web navigation for wireless devices. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 879–884, Seattle, WA, 2001. Morgan Kaufmann.

[Anderson *et al.*, 2002] C. Anderson, P. Domingos, and D. Weld. Relational Markov models and their application to adaptive Web navigation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 143–152, Edmonton, Canada, 2002.

[Boutilier *et al.*, 2001] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDPs. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 690–697, Seattle, WA, 2001. Morgan Kaufmann.

[Chakrabarti *et al.*, 1998] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 307–318, Seattle, WA, 1998. ACM Press.

[Doan *et al.*, 2001] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 509–520, Santa Barbara, CA, 2001. ACM Press.

[Doan *et al.*, 2002] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the Semantic Web. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 662–673, Honolulu, HI, 2002. ACM Press.

[Doan *et al.*, 2003a] A. Doan, P. Domingos, and A. Halevy. Learning complex semantic mappings between structured representations. 2003. Submitted.

[Doan *et al.*, 2003b] A. Doan, P. Domingos, and A. Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50:279–301, 2003.

[Domingos and Hulten, 2000] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, Boston, MA, 2000. ACM Press.

[Domingos and Richardson, 2001] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66, San Francisco, CA, 2001. ACM Press.

[Doucet *et al.*, 2001] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York, 2001.

[Dzeroski, 1996] S. Dzeroski. Inductive logic programming and knowledge discovery in databases. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 117–152. AAAI Press, Menlo Park, CA, 1996.

[Friedman *et al.*, 1998] N. Friedman, D. Koller, and A. Pfeffer. Structured representation of complex stochastic systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 157–164, Madison, WI, 1998.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1300–1307, Stockholm, Sweden, 1999. Morgan Kaufmann.

[Hulten and Domingos, 2002] G. Hulten and P. Domingos. Mining complex models from arbitrarily large databases in constant time. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 525–531, Edmonton, Canada, 2002. ACM Press.

[Hulten *et al.*, 2003] G. Hulten, P. Domingos, and Y. Abe. Mining massive relational databases. In *Proceedings of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, Acapulco, Mexico, 2003. This volume.

[Jensen and Neville, 2002a] D. Jensen and J. Neville. Autocorrelation and linkage cause bias in evaluation of relational learners. In *Proceedings of the Twelfth International Conference on Inductive Logic Programming*, Sydney, Australia, 2002. Springer.

[Jensen and Neville, 2002b] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 259–266, Sydney, Australia, 2002. Morgan Kaufmann.

[Kautz *et al.*, 2003] H. Kautz, O. Etzioni, D. Fox, D. Weld, and L. Shastri. Foundations of assisted cognition systems. Technical Report CSE-03-AC-01, Department of

Computer Science and Engineering, University of Washington, Seattle, WA, 2003.

[Kersting *et al.*, 2003] K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards discovering structural signatures of protein folds based on logical hidden Markov models. In *Proc. 8th Pacific Symposium on Biocomputing*, Kauai, HI, 2003.

[Kersting, 2000] K. Kersting. *Bayesian Logic Programs*. PhD thesis, University of Freiburg, Freiburg, Germany, 2000.

[Koller and Pfeffer, 1997] D. Koller and A. Pfeffer. Learning probabilities for noisy first-order rules. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 1316–1321, Nagoya, Japan, 1997. Morgan Kaufmann.

[Lau *et al.*, 2003a] T. Lau, P. Domingos, and D. Weld. Learning programs from traces using version space algebra. 2003. Submitted.

[Lau *et al.*, 2003b] T. Lau, S. Wolfman, P. Domingos, and D. Weld. Programming by demonstration using version space algebra. *Machine Learning*, 2003. To appear.

[Mausam and Weld, 2003] Mausam and D. Weld. Solving relational MDPs with first-order machine learning. In *Proceedings of the ICAPS-2003 Workshop on Planning under Uncertainty and Incomplete Information*, Seattle, WA, 2003.

[McCallum *et al.*, 1998] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 359–367, Madison, WI, 1998. Morgan Kaufmann.

[Murphy and Russell, 2001] K. Murphy and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 499–516. Springer, New York, 2001.

[Ngo and Haddawy, 1997] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.

[Perkowitz and Etzioni, 1997] M. Perkowitz and O. Etzioni. Adaptive Web sites: An AI challenge. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 16–21, Tokyo, Japan, 1997. Morgan Kaufmann.

[Richardson and Domingos, 2002a] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in PageRank. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1441–1448. MIT Press, Cambridge, MA, 2002.

[Richardson and Domingos, 2002b] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 61–70, Edmonton, Canada, 2002. ACM Press.

[Richardson and Domingos, 2003a] M. Richardson and P. Domingos. Building large knowledge bases by mass collaboration. 2003. Submitted.

[Richardson and Domingos, 2003b] M. Richardson and P. Domingos. Learning with knowledge from multiple experts. 2003. Submitted.

[Richardson *et al.*, 2003] M. Richardson, R. Agrawal, and P. Domingos. Building the Semantic Web by mass collaboration. 2003. Submitted.

[Sanghai *et al.*, 2003] S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003. Morgan Kaufmann.

[Smyth *et al.*, 1997] P. Smyth, D. Heckerman, and M. I. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9:227–269, 1997.

[Yoon *et al.*, 2002] S. Yoon, A. Fern, and R. Givan. Inductive policy selection for first-order Markov decision processes. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, Edmonton, Canada, 2002. Morgan Kaufmann.

Social Network Relational Vectors for Anonymous Identity Matching

Shawndra Hill
NYU Stern School of Business
44 W 4th Street
New York, NY 10012
shill@stern.nyu.edu

Abstract

Anonymous fraudulent behavior can generate substantial financial burden and inconvenience. Moreover, the recent threat of terrorist infiltration to both business and government has yielded heightened interest in anonymous identity matching (AIM). Most applications of AIM require sophisticated methods robust to issues such as deliberate variation in identity attributes, missing data, and multi-source data corpora. We consider relational social network behavior, eliminating the reliance on personal identifiable data for identity matching. In particular, we consider problems that can be characterized by personal communication networks. We evaluate a proposed implementation of a social network vector-space relational model for AIM on CiteSeer, a research publication citation database.

1 Introduction

AIM has garnered attention by government agencies in the wake of perceived increased domestic asymmetric threat. However, civilians are concerned about the potential exploitation of data collected by government agencies, web enabled click stream technologies, credit card companies, and health care providers. The ongoing debate surrounding the tension between security and privacy has motivated data mining research in data privacy.

At the forefront of data mining privacy research are methods that solely rely on perturbed datasets while maintaining predictive performance of various modeling techniques (Agrawal and Srikant 2000; Clifton 2000; Agrawal and Aggarwal 2001) (Atallah, Bertino et al. 1999). More relevant to the AIM discussion are database inference techniques which utilize multi-source data, (Moskowitz 1999; L. Sweeney 2002) to identify individuals who otherwise could not be categorically linked using isolated data sources.

In general, privacy research considers three distinct categories: 1) basic storage and retrieval, i.e., who can access sensitive data; 2) pattern discovery, i.e., the misuse of sensitive data for pattern discovery; and 3) combination of group patterns, i.e., who can make inferences about individual identity from aggregated data sources (Piatetsky-Shapiro 1995). Despite efforts to encrypt sensitive information, this research indicates that relationship networks may be a subtle indicator of identity.

In this paper, we consider a straightforward method, a social network (Wasserman and Faust 1994; Scott 2000) vector-space model, for AIM in networks of interpersonal relationships. Social network analysis is an appropriate basis for relational learning because: (1) it quantifies relationships; (2) it is well defined; (3) it can be used as a complement to other methods; and (4) it can be used for visualization to enable further understanding of underlying phenomena.

An *actor* is the social entity of interest in a social network. Actors are discrete individuals, or collective social units. In our context, actors may be individuals, companies, industries or nations; we first consider one-mode networks where the actors are considered the same type. A *relational tie* establishes a linkage between a pair of actors. Examples of relational ties include online communication, business transactions, belonging to the same professional club, or a physical/virtual connection. Each actor pair relationship is given a *weight* to indicate the strength. Each actor may have multiple relationships to multiple actors. A vector of weights then represents each actor.

This paper considers research in progress on AIM. We demonstrate the usefulness of the social network vector-space relational model on the application of author identification. The paper is organized as follows. First, we present the social network vector-space model in section 2. In section 3 we apply the vector-space model to the task of author identification and present preliminary results on the CiteSeer database. Finally, we conclude by offering a discussion of results and future research directions in section 4 and 5 respectively.

2 Method

For AIM, we would like to classify new relational examples given a set of labeled relational training examples. We consider social network graphs of relationships by

reducing the social network relational graphs to feature vectors of entities. Each new entity in turn represents a candidate example for identification. Weighted term vectors represent all individual entities.

Definition: An entity e_i can be described by an entity vector,

$$e_i = (w_{i1}, w_{i2}, \ldots, w_{il}) \quad [1]$$

where w_{ik} is the weight assigned to the entity e_k in entity relationship e_i.

The feature vectors of entities are weighted to give emphasis to stronger entity pair relationships. The weight is determined by the aggregation of all relationships between two entities.

Furthermore, one can specify to what distance in the graph, related entities are considered. At distance one, entities simply represent the weighted adjacency matrix of the relationship graph. At greater distances, however, the entity is composed of embedded entities. To consider entities that embed distant entities, each entity is recursively joined with each of its related entities.

Definition: Under addition, entity E is defined by the weighted union of all nodes (entities), and edges (weights) in e_1 and e_2 where α and β are scalars.

$$E = \alpha e_1 \oplus \beta e_2 \quad [2]$$

The scalars are utilized to indicate relative significance to the resultant entity. For example, one may want to decay the impact of joined edges in the relationship vector as the distance from the node in the graph increases.

The weight of an edge in E is therefore defined by [3].

$$w(E) = \alpha w(e_1) \oplus \beta w(e_2) \quad [3]$$

Definition: An entity that takes into consideration relational links of distance greater than one may therefore be defined as the recursive sum of each entity with its feature vector entities e_k.

$$E = \bigoplus_{i=1}^{k} e_i \quad [4]$$

During the AIM process, new entities are compared to labeled entity vectors. Candidate match sets of entity vectors closest to the query considered similar are ranked and returned. For this exposition, we measure similarity by the cosine distance between the corresponding vector pairs [5]. However, any vector based similarity measure may be considered. The distance measures may be used in standard hierarchical clustering techniques such as dendrograms (Duda, Hart et al. 2001).

$$dist(q_j, e_k) = \frac{\vec{q}_j \cdot \vec{e}_k}{|\vec{q}_j||\vec{e}_k|} = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,k}^2}} \quad [5]$$

3 Experiment

Many refereed journals maintain that anonymity in publication submission is an ethical prerequisite of paramount importance. Nonetheless, we find that reference lists alone identify authors remarkably well. This experiment considers the question of whether the author of a new paper can be identified utilizing solely the citation graph of the paper. We apply the social network vector-space relational model to the CiteSeer database (Lawrence 1999), a scientific literature digital library. We identify authors of papers published in the year 2000 by considering only their citation graph.

Prior Pubs	2000 Pubs	<2000 Pubs	2000 Authors
10	13,174	93,831	8,615
20	9,405	68,597	3,334
30	6,797	50,294	1,659
40	4,678	35,223	855
50	3,462	26,010	510
60	2,636	19,158	315
70	1,932	13,827	191
80	1,540	10,461	128
90	1,201	8,118	91
100	852	5,777	59

Table 1: CiteSeer Data: *Distribution of papers authored and authors with at least n prior publications.*

First, background knowledge is constructed using prior publication knowledge. For each document published prior to 2000 an edge is created linking each author to each cited author. A weighted vector of cited authors defines an author. Next, weighted adjacency vectors are created for each document in 2000. An edge is created between the document and each cited author. A weighted vector of cited authors defines each document. The weights are defined as the total sum of out going links for each author-author, document-author pair.

For this experiment, we are interested in exact author matches as opposed to finding similar authors. Therefore, we limit the dataset to include only papers authored by writers with publication history. The number of prior publications in the background knowledge database determines publication history. We present results for different levels of publication history [Table1] to

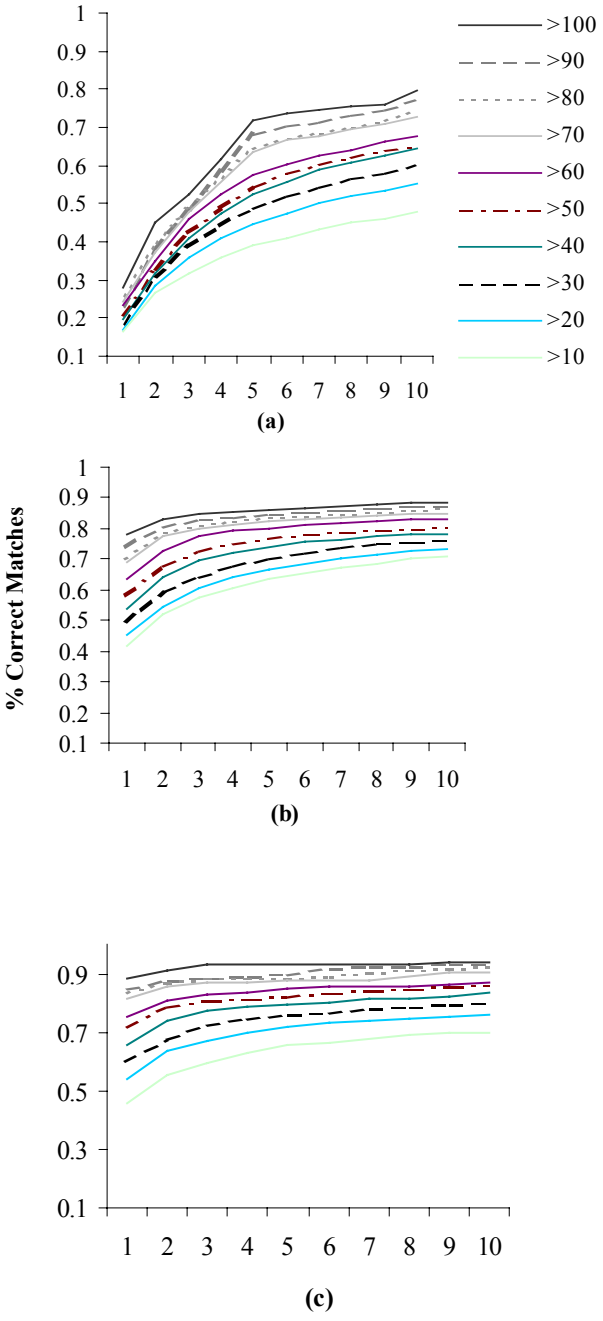


Figure 1: Author Match Success: *Observed proportions of author matches versus the set size of ranked candidate matches. Each line represents results for documents whose authors had greater than n publications prior to the year 2000, (a) documents compared to entire CiteSeer database and documents compared to data set segmented by publication history (b) without and (c) with a filter for strength >1*

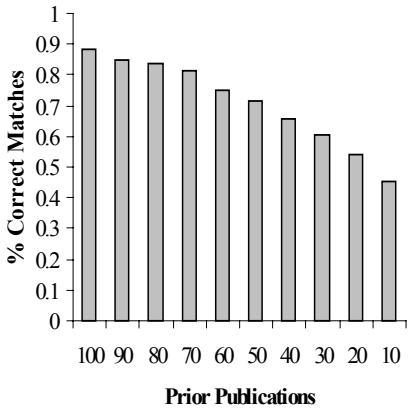


Figure 2: Author Match Success. *Observed proportions of author matches for top ranked candidate versus prior publication record of at least one author.*

understand further attributes that may influence identity matches in large relational networks.

We present results on three experiments motivated by subscription fraud in relational networks (Cortes, Pregibon et al. 2002). We use author identification in the CiteSeer database as a proxy problem to subscription fraud detection.

First, subscription fraud cases generate financial burden to organizations when left undetected. Therefore, the most prudent of methods generate risk scores for all subscribers. Each potential subscriber is compared to the knowledge base of all known subscription fraud offenders before services are rendered. As such, we consider matching documents in 2000 to the entire historical database.

We find that authors with more than 10 prior publications can be identified with 17% accuracy (recall that this is from a total of 8615 authors); 28% of the time the true author is in the top-10 candidate matches. Authors with more than 90 prior publications can be identified with 58% accuracy; 80% of the time the true author is among the top-10 candidates [Figure 1a].

Second, subscription fraud often considers "guilt by association". In this case, new subscribers are compared only to a subset of fraudulent entities in the knowledge base population that are related in some way. We model this problem by considering sub samples of the knowledge base corresponding to publication history.

We find that authors with more than 10 prior publications can be identified with 78% accuracy; 41% of the time the true author is in the top-10 candidate matches. Authors with more than 90 prior publications can be identified with 87% accuracy; 71% of the time the true author is among the top-10 candidates [Figure 1b]. .

Finally, a naïve method to remove uninformative edges is to limit the citation graph by considering only relationships with relatively high strength greater than *n* [Figure 1c]. The % Correct Matches significantly increased by refining

our search. For authors with more than 10 prior publications, we compared 13,174 documents to 93,831 documents with 8,615 authors [Table 1] and yielded 45.6 % matches to the top ranked candidate [Figure 2]. In practice, filters are used to reduce the time and space complexity of identity retrieval techniques.

In summary, we first show that our simple method works for author matches under different conditions for both the knowledge base and test set. In an attempt to further refine our search and reduce noise in the knowledge base, we followed with an experiment utilizing smaller samples of the knowledge base segmented by publication history. This task refinement resulted in a significant increase in author identification from 28% to 41% in task accuracy with more than 10 prior publications. Finally, we attempt to reduce noise in the test set by filtering less informative nodes which in turn yield accuracy of 45 % on authors with greater than 10 publications in the past. It is important to note that results are shown for different candidate set sizes because in practice, human operators often have the ability to work multiple cases.

4 Discussion

In this paper, we introduce the concept of social network vector space model for anonymous identity matching. We concentrate on the method and show preliminary results on a real world citation database.

Our results indicate that considering the network structure of author's reference list does remarkably well at identifying authors, and combining the social network vector-space model with (for example) linguistic analysis may perform even better (Khmelev 2000).

If we can further understand relationships between research community/author network identification and fraud detection, we may inform subscription fraud identification techniques with our method where test labels are abundantly available.

5 Future Research

There are many interesting challenges, to behavioral AIM. First, personal communication networks are dynamic and require data structures (Cortes, Pregibon et al. 2002) that capture network evolution through time. Furthermore, the strength of a relationship may not always be determined by absolute frequency. A less "frequent" relationship may be a stronger indication of identity. In general, communication networks are large but sparse. Techniques are needed to preserve graph structure while reducing dimensionality. AIM techniques must consider that communication networks are inherently noisy, fraudulent individuals for example may either attempt to hide their identity or steal that of someone else. Finally, evaluation methods are needed to assess unlabelled anonymous entities matches

The research synopsis considers research in progress. In the future, we will consider multi-attribute entity relationships in our model. We will consider complementing the AIM ranking with available identifiable actor information. In addition, we plan to add linguistic analysis attributes to our relationship vector in the future for author identification. We want to further develop a data structure that will incorporate the dynamic nature of communication links.

We will compare and contrast AIM results of other vector space models such as naïve Bayes, information retrieval TF-IDF, and support vector machines. Furthermore, we will demonstrate the efficacy of the proposed method to other communication network domains such as web logs, email logs, long distance calling records and prepaid calling card records. Finally, we plan to investigate appropriate evaluation methodologies where test labels are non-existent.

Acknowledgements

Thanks to Foster Provost for valuable comments on this draft and continued research support. Daryl Pregibon and Corinna Cortes for research funding in multi-relational data mining. Data used in this paper were drawn from the CiteSeer database (http://citeseer.nj.nec.com/). This work is sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-01-2-585.

References

Agrawal, D. and C. Aggarwal (2001). On the design and quantification of privacy preserving data mining algorithms. In Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Santa Barbara, California, USA, ACM.

Agrawal, R. and R. Srikant (2000). Privacy-preserving data mining. In Proceedings of of the 2000 ACM SIGMOD International Conference on Management of Data.

Atallah, M. J., E. Bertino, et al. (1999). Disclosure limitation of sensitive rules. Proceedings of IEEE Knolwedge and Data Engineering Workshop, Chicago, Il.

Clifton, C. (2000). "Using Sample Size to Limit Exposure to Data Mining." Journal of Computer Security **8**(4).

Cortes, C., D. Pregibon, et al. (2002). Communities of Interest for Dynamic Graphs. In the Proceedings of Knowlege Discovery and Data Mining Conference, Edmonton, Canada.

Duda, R. O., P. E. Hart, et al. (2001). Pattern classification. New York, Wiley.

Khmelev, D. V. (2000). "Disputed Authorship Resolution through Using Relative Empirical Entropy for Markov Chains of Letters in Human Language Texts." Journal of Quantitative Linguistics **7**(3): 201-207.

L. Sweeney, L. (2002). "K-anonymity: A model for protecting privacy. International." Journal on Uncertainty, Fuzziness, and Knowledge-based Systems **10**(7): 557-570.

Lawrence, S., Giles, L., Bollacker, K (1999). "Digital Libraries and Autonomous Citation Indexing NEC Research Institute." IEEE Computer **32**(6): 67-71.

Moskowitz, L. C. a. I. S. (1999). Parsimonious Downgrading and Decision Trees Applied to the Inference Problem. The Workshop of New Security Paradigms.

Piatetsky-Shapiro, G. (1995). "Knowledge Discovery in Personal Data versus Privacy---A Mini-Symposium."

Scott, J. (2000). Social network analysis : a handbook. London ; Thousand Oaks, Calif., SAGE Publications.

Wasserman, S. and K. Faust (1994). Social network analysis : methods and applications. New York, Cambridge University Press.

Mining Massive Relational Databases

Geoff Hulten, Pedro Domingos, and Yeuhi Abe
Department of Computer Science and Engineering
University of Washington, Seattle, WA, 98195-2350
{ghulten, pedrod, yeuhi}@cs.washington.edu

Abstract

There is a large and growing mismatch between the size of the relational data sets available for mining and the amount of data our relational learning systems can process. In particular, most relational learning systems can operate on data sets containing thousands to tens of thousands of objects, while many real-world data sets grow at a rate of millions of objects a day. In this paper we explore the challenges that prevent relational learning systems from operating on massive data sets, and develop a learning system that overcomes some of them. Our system uses sampling, is efficient with disk accesses, and is able to learn from an order of magnitude more relational data than existing algorithms. We evaluate our system by using it to mine a collection of massive Web crawls, each containing millions of pages.

1 Introduction

Many researchers have found that the relations between the objects in a data set carry as much information about the domain as the properties of the objects themselves. This has lead to a great deal of interest in developing algorithms capable of explicitly learning from the relational structure in such data sets. Unfortunately, there is a wide and growing mismatch between the size of relational data sets available for mining and the size of relational data sets that our state of the art algorithms can process in a reasonable amount of time. In particular, most systems for learning complex models from relational data have been evaluated on data sets containing thousands to tens of thousands of objects, while many organizations today have data sets that grow at a rate of millions of objects a day. Thus we are not able to take full advantage of the available data.

There are several main challenges that must be met to allow our systems to run on modern data sets. Algorithmic complexity is one. A rule of thumb is that any learning algorithm with a complexity worse than $O(n \log n)$ (where n is the number of training samples) is unlikely to run on very large data sets in reasonable time. Unfortunately, the global nature of relational data (where each object is potentially related to every other object) often means the complexity of relational learning algorithms is considerably worse than this. Additionally, in some situations for example when learning from high speed, open ended data streams even $O(n)$ algorithms may not be sufficiently scalable. To address this, the most scalable propositional learning algorithms (for example BOAT [Gehrke *et al.*, 1999] and VFDT [Domingos and Hulten, 2000]) use sampling to decouple their runtimes from the size of training data. The scalability of these algorithms depends not on the amount of data available, but rather on the complexity of the concept being modeled. Unfortunately, it is difficult to sample relational data (see Jensen [1998] for a detailed discussion) and these propositional sampling techniques will need to be modified to work with relational data. Another scaling challenge is that many learning algorithms make essentially random access to training data. This is reasonable when data fits in RAM, but is prohibitive when it must be repeatedly swapped from disk, as is the case with large data sets. To address this, researchers have developed algorithms that carefully order their accesses to data on disk [Shafer *et al.*, 1996], that learn from summary structures instead of from data directly [Moore and Lee, 1997], or that work with a single scan over data. Unfortunately, it is not directly clear how these can be applied in relational settings. Another class of scaling challenges comes from the nature of the processes that generate large data sets. These processes exist over long periods of time and continuously generate data, and the distribution of this data often changes drastically as time goes by.

In previous work [Hulten and Domingos, 2002] we developed a framework capable of semi-automatically scaling up a wide class of propositional learning algorithms to address all of these challenges simultaneously. In the remainder of this paper we begin to extend our propositional scaling framework to the challenge of learning from massive relational data sets. In particular, we describe a system, called VFREL, which can learn from relational data sets containing millions of objects and relations. VFREL works by using sampling to help it very quickly identify the relations that are important to the learning task. It is then able to focus its attention on these important relations, while saving time (and data accesses) by ignoring ones that are not important. We evaluate our system by using it to build models for predicting the evolution of the Web, and mine a data set containing over a million Web pages, with millions of links among them.

In the next section we describe the form of the relational data our system works with. Following that we briefly review some of the methods currently used for relational learning and discuss the challenges to scaling them for very large data sets. The following section describes VFREL in detail. We then discuss our application and the experiments we conducted, and conclude.

2 Relational Data

We will now describe the form of the relational data that we mine. This formulation is similar to those given by Friedman *et al.* [1999] and by Jensen and Neville [2002c]. Data arrives as a set of *object sources*, each of which contains a set of *objects*. Object sources are typed, and thus each is restricted to contain objects conforming to a single *class*. It may be helpful to think of an object source as a table in a relational database, where each row in the table corresponds to an object. In the following discussion we will use X to refer to an object and $C(X)$ to refer to its class. Each class has a set of *intrinsic attributes*, and a set of *relations*. From these, a set of *relational attributes* is derived. We will describe each of these in turn.

Intrinsic attributes are properties of the objects in the domain. For example a *Product* object's attributes might include its price, description, weight, stock status, etc. Each attribute is either numeric or categorical. We denote the set of intrinsic attributes for $C(X)$ as $A(C(X))$ and X's intrinsic attribute named a as $X.a$.

Objects can be related to other objects. These relations are typed, and each relation has a source class and a destination class. *Following a relation* from an instance of the source class yields a (possibly empty) set of instances of the destination class. One critical feature of a relation is the cardinality of the set of objects that is reached by following it. If a relation always returns a single object it is called a *one-relation*; if the number of objects returned varies from object to object it is called a *many-relation*. Our notation for a relation r on class $C(X)$ is $C(X) \rightarrow r$. We denote the set of relations for $C(X)$ as $REL(C(X))$. We will use $X \Rightarrow r$ to denote the set of objects reached by following relation r from object X, and we will use $C(X) \Rightarrow r$ to denote the target class of the relation. The series of relations that are followed to get from one object to another is called a relational path. Also note that every relation has an inverse relation. For example, the inverse to the *Product* $\rightarrow$ *producedBy* relation is the *Manufacturer* $\rightarrow$ *produces* relation.

An object's *relational attributes* are logical attributes that contain information about the objects it is related to. For example, one of a *Product* object's relational attributes is the total number of products produced by its manufacturer. Relational attributes are defined recursively, and the relational attributes of an object consist of the intrinsic attributes and relational attributes of the objects it is related to, and so on. It is common to limit the depth of recursion in some manner.

Each object must have a fixed number of relational attributes for any given depth to facilitate the use of existing tools on relational data. Unfortunately each object with many-relations (or that is related to an object with many-relations) has a variable number of related objects for any given depth. In order to reconcile this difference, we aggregate the values of a set of instances into a fixed number of attributes using a set of aggregation functions. The attributes for any particular instance are a subset of the attributes that are possible at a class level (if a many-relation on an instance is empty, some of the class level attributes have no value for the instance). Thus, more formally, let $R(C, d)$ be the set of relational attributes for C up to a depth of d. Let the set of all attributes (intrinsic and relational) for the class to depth d be $ATT(C, d) = A(C) \cup R(C, d)$.

$$R(C,d) = \bigcup_{r \in REL(C)} \bigcup_{a \in ATT(C \Rightarrow r, d-1)} AGG(a) \quad (1)$$

When r is a one-relation AGG is the identity function. When r is a many-relation AGG applies a set of aggregation functions to a and results in one attribute per aggregation function. The aggregations used depend on the type of a; in our experiments we use min, max, mean, and standard deviation if a is numeric and mode if a is categorical. We also include one additional relational attribute per many-relation, which is the count of the number of objects that satisfy the relation. Each relational attribute uses an intrinsic attribute from a single class, and passes it through the set of aggregation functions for each many-relation between $C(X)$ and the class with the intrinsic attribute. For example, the relational attributes of *Manufacturer* might include the average price of products it produces, the maximum price of a product it produces, the count of the number of products it produces, the most common color of a product it produces, the maximum of the average sale price of products it produces, etc.

The definition of R above is at the class level, but we are interested in the values for these attributes at an instance level. This is simply a matter of starting from the instance, following relations and calculating aggregations as specified in the preceding definition. We describe this procedure in more detail (including pseudo-code) in Section 4.1.

3 Relational Learning

One of the possible goals for relational learning is to build models to predict the value of some *target attribute* (or attributes) of a *target class* (or classes) from the other attributes of the objects of the target class and the objects they are related to. (Note that the target attribute can be intrinsic or relational.) A training data set–with the values of the target attributes filled in–is presented to the learner, and the learner must produce a model that accurately predicts the values of the target attributes on some other data set, where they are unknown. This is the type of relational learning we will focus on in the remainder of this paper. Other possible goals for relational learning systems include building probabilistic models over link existence and object existence (see Getoor *et al.* [2001]).

Perhaps the simplest method for performing relational learning is to *flatten* the data into a propositional data set, and pass it to an existing propositional learning system. Flattening proceeds as follows: a target depth d is selected, and a

propositional training example is constructed for each object in the target source by calculating the values of the attributes in the set $ATT(C(X), d)$. The advantage of this method is its simplicity, but it has several disadvantages. One is that it is very slow: calculating the value for each attribute potentially requires loading a large portion of data set from disk, and, even for modest values of d, there can easily be thousands or tens of thousands of attributes for each object. This problem grows worse than linearly with the size of the relational data set, because larger data sets have more objects that need their attributes calculated, and each of these objects is related to more objects in the larger data set; the exact cost depends on the density of the relational structure in the data. Another disadvantage of this method is that it produces propositional learning problems with very large attribute spaces. Large attribute spaces lead many learning algorithms to overfit the training data. Further, this often means that the size of the flattened data set is much larger than the relational one, which leads to additional scaling challenges.

One method used to address these problems is to avoid flattening the entire database, and instead perform a search over the space of possible relational attributes. This is the method used by PRMs [Friedman *et al.*, 1999]. PRMs work by first selecting a small subset of the possible attributes using some form of feature selection. Sufficient statistics are gathered for the selected attributes and passed to a propositional learner (PRMs use a Bayesian Network learning algorithm, modified to learn coherent joint distributions in the presence relational data). When the learner produces a model, a new set of attributes is selected by performing another round of feature selection, taking into account the information contained in the partially learned model. The system gathers any new sufficient statistics it needs, and the propositional learner is called to refine its existing model with the new set of attributes. These steps are repeated until resources are exhausted or until the quality of the resulting model asymptotes.

These approaches improve on flattening, but they still do not scale to very large data sets. One reason is that they must flatten each attribute they are considering for every object in the target source before they can do any feature selection. This is wasteful because the feature selection task is often relatively easy, and decisions could be made much sooner with high confidence. Additionally, the greedy search procedures they use may miss interesting features that could be easily found with more systematic search. In the next section we will present our system, which addresses these problems.

4 Scaling Up Relational Learning

Our system, which we call VFREL, has three main components. The first is a query planner designed to provide efficient access to training data on disk as needed by the rest of our system. The second is a filter-based feature selection algorithm that is accelerated with sampling. The third is a propositional learning algorithm. At a high level, VFREL works by using sampling to select a promising subset of the possible relational attributes, saving time by flattening those while ignoring the others, and then calling a propositional learner on the flattened values. In particular, it begins by scanning a sample of the target objects and flattening all attributes up to a large depth. This is very slow, but VFREL only does it for a small sample of the target objects. It then pauses and uses statistical tests to identify attributes that are poor enough that, with high confidence, they would not be selected by the feature selection algorithm if it could see their values for all of the target objects. As soon as it identifies any such clear losers, VFREL saves time by eliminating them from further consideration. VFREL repeats this procedure, generating fewer and fewer attribute values (requiring fewer disk accesses and less processor time) on more and more of the data set. After computing attribute values for all of the target objects, VFREL performs a final round of feature selection, constructs a propositional data set from the final set of selected attributes, and passes it to a propositional learning algorithm. We will now describe the components of our system in more detail, starting with our data access module.

4.1 Efficient Data Access: Traversal Tree

VFREL needs to calculate the values of some relational attributes for each target object. In order to do this, every related object that is relevant to one of these attributes needs to be loaded from disk and processed. VFREL can determine which relations it needs to follow to gather this set of objects from the information it has at class level. It builds a tree of these required relational paths. It then traverses the tree, following each relation at most once, loading data into RAM only as it is needed, and computing the required attribute values. Traversal Trees work with binary relations. If the domain contains N-ary relations, they are encoded into binary relations in the usual way.

Nodes in the traversal tree correspond to classes, and edges correspond to relations. During its run, VFREL maintains a tree that contains exactly the relations that must be followed to calculate the values of the relational attributes of the target object that have not been determined to be clear losers. And so, at each node in the tree, VFREL maintains a list of the attributes whose values need to be calculated from the instances of that class that are encountered at that point in a traversal. In VFREL's first iteration the traversal tree is simply an unrolled version of the class graph, and can be constructed in time proportional to the size of $ATT(TargetClass, d)$ as follows. The root node represents the class of the target object. A child is added to this node for each class in $REL(TargetClass)$, and so on recursively until the tree is depth d. Let T be a node, T_c be the class represented by the node, e be an edge, and e_r be the relation represented by the edge. Next, we build a list on each node (let T_a be the list on node T) that represents the attribute values that must be calculated at that point in the traversal as follows. We compute the set $ATT(TargetClass, d)$. Each of these attributes is based on one of the attributes of one class (see Equation 1) and is added to the associated node's list. Following cycles in the relational structure can lead to some obviously redundant attributes. Many such attributes are removed at this point by removing length one cycles that involve a one-relation and its inverse.

When VFREL needs to calculate the value of the relational attributes for a particular target object it uses the traversal tree

Table 1: Pseudo-code for calculating attribute values with a traversal tree.

```
Procedure Traverse(T, O)
  T is a traversal tree
  O is an instance of class T_c
  Let R = {} be the results of the traversal
  Record in R the values for attributes in T_a from O
  For e ∈ Children(T)
    Let T^ch be the node reached via e
    Let Objs be X ⇒ e_r
    If Objs is empty, every attribute in T^ch and all of
      its children is missing, note this in R
    If e_r is a one relation, let O^ch be the object in Objs
      R = R ∪ Traverse(T^ch, O^ch)
    Else e_r is a many relation, let Tmp = {}
      For O^ch ∈ Objs
        Tmp = Tmp ∪ Traverse(T^ch, O^ch)
      Perform needed aggregations, note values in R
  Return R
```

to determine which objects to load from disk and when. Table 1 contains pseudo-code for the procedure.

As the run progresses, and attributes are eliminated by feature selection, VFREL will remove the eliminated attributes from the attribute lists on the traversal tree's nodes. Notice that a leaf with an empty attribute list corresponds to an object where every attribute has been determined to be a loser. Such objects do not need to be loaded from disk and so the leaf is pruned from the tree (internal nodes may have empty lists as they can still contribute through the objects that they are related to).

This traversal strategy allows VFREL to follow each edge in the traversal tree only once (instead of once per attribute, as might be done if following an edge required just a pointer dereference instead of a disk access).[1] It also allows VFREL to be very efficient with its RAM usage. In particular, at any point in the traversal it requires that one object be in RAM per edge in the path from the root to the current traversal tree node. It also requires RAM to store the partially computed attribute values. (The maximum space required for this is on the order of the number of relational attributes of the target class, since relational attribute values are computed in an online manner as objects are loaded from disk.) For each many-relation VFREL also maintains a list of hash keys for the objects it will need to load to finish following the relation.

4.2 Feature Selection with Sampling

Our system uses filter-based feature selection [Kononenko, 1994], [Kohavi and John, 1997] to explore the space of relational attributes. The goal is to identify the relational attributes that are most relevant to the learning task and accelerate our system by only calculating the values of these relevant attributes, while ignoring the rest. VFREL uses sampling to accelerate this process, and is able to eliminate attributes (and thus paths from the traversal tree) with less than one scan over the data set. This allows it to be more efficient than standard PRM learning.

Filter-based feature selection works as follows. The utility of each feature is estimated on training data with a scoring function (commonly information gain). The best N features are selected, and the rest are discarded. Traditionally, calculating the information gain of an attribute requires knowing the value of the attribute for every training example. In our context, this means that the entire data set needs to be flattened before feature selection can take place, which results in no speed gain. If we are willing to accept a small chance of making an error, we can use sampling to do much better. VFREL uses techniques developed by Hulten and Domingos [2002] and others to do just that. Standard statistical results can be used to obtain a high confidence bound on the difference between the gain observed for a feature on a sample of data and the true gain of the feature. For example, the Hoeffding bound [Hoeffding, 1963] says the following. Let x be a random variable with range R. Let $\hat{x}$ be the mean of n i.i.d. (independent and identically distributed) observations of x. Then, with probability $1-\delta$, the Hoeffding bound guarantees that $x > \hat{x} - \epsilon$ where

$$\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}} \tag{2}$$

We apply this bound to our setting as follows. Let $G(A_1, n)$ be the information gain observed for attribute A_1 on a sample of n examples and similarly for $G(A_2, n)$. Recall that the range of the information gain function is the log base two of the number of values of the target attribute. Let $\Delta = G(A_1, n) - G(A_2, 2)$. We bound Δ with the Hoeffding bound and thus, if $\Delta - \epsilon > 0$, we know with confidence $1-\delta$ that A_1 truly has a higher information gain than A_2, and thus that the feature selection algorithm would select A_1 over A_2 if the gains were computed from the entire training set, instead of from the sample. Thus, when trying to find the top N features in the training set, and after the values of relational attributes have been generated for the first n target objects, we can state the following. Let A_N be the attribute with the N^{th} best gain on the sample. Then, with confidence $1-\delta^*$, any attribute with a gain less than $G(A_N, n) - \epsilon$ is not one of the best N attributes. δ^* is different from the δ in the Hoeffding bound because many comparisons are involved in the feature selection, and thus the bound needs to be applied many times to assure a global level of confidence. We use a Bonferroni correction and set δ by dividing δ^*, the desired global confidence, by the number of bounds that need to hold during the algorithm's entire run.

Sampling from relational data may violate the i.i.d. requirement of the Hoeffding bound. Taking this into account, using non-i.i.d. extensions of Hoeffding-style bounds, is an important direction for future research (see also Jensen and Neville [2002a] [2002b]).

[1] Notice that the description here may require an object be loaded from disk more than once per traversal if it is reached via several different relational paths. The full VFREL system uses several forms of caching to reduce this redundancy, but they are not reported on or evaluated in this paper.

Table 2: Pseudo-code for the VFREL algorithm.

Let $F = ATT(TargetClass, d)$
Let T = Initial Traversal tree for F
Let D = Initial step size
Let C = Database cursor for the target object source
While C is not finished
  Calculate values for F on next D objects from C
  Compute information gain for attributes in F
  Order F by information gain
  Let G_N be gain of the N^{th} best attribute
  Remove from F every attribute with gain $< G_N - \epsilon$
  Update T by dropping the removed attributes
  Call the StepSize function to find next D

Return the result of the propositional learner on the
  best N attributes

4.3 The VFREL Algorithm

We will now describe VFREL, our algorithm for mining massive relational data sets, in detail. The inputs are a relational data set, a target class and target attribute of that class, a depth cutoff d, a global confidence target δ^*, a target number of features N, a function that specifies how many samples to take before performing a round of feature selection (StepSize below), and a propositional learning algorithm. Table 2 contains pseudo-code for VFREL.

VFREL iterates over the target objects and starts generating values for all of the attributes that are at most depth d away. It periodically pauses to perform a round of feature selection, informed by the data that has been generated up to that point. The information gain for each of the attributes being considered is computed, and they are sorted by their information gain. The N^{th} best attribute is determined, and its information gain is noted. From the Hoeffding bound, we know with high probability that any attribute with a score less than $G_N - \epsilon$ will not be selected as one of the final N attributes, and does not need to be considered further. In order to assure a global confidence of δ^* that the correct attributes are selected, each local ϵ is determined with $\delta = \delta^*/(|ATT(TargetClass, d)| * i)$, where i is an estimate of the total number of iterations of VFREL's main loop that will be performed[2]. When VFREL finishes with all the target objects, it performs one final round of feature selection, keeping only the top N features. Finally, a propositional data set is created from the attribute values that were calculated during the feature selection and the propositional learning algorithm is called to produce a model.

Notice that this algorithm assumes that objects are retrieved from the target object source in random order, which is usually possible. In our application, for example, we iterate over the keys of a DBM style hash table, which returns keys in essentially random order. Other settings may require a scan over the data set to randomize it.

Early iterations of VFREL take relatively long, as they generate values for many attributes, and thus require many objects be loaded from disk. As the algorithm proceeds, however, it is able to eliminate attributes that are clearly not going to be selected, stop following the relations associated with them, focus its attention on the promising attributes, and thus generate attribute values for later object much more quickly. VFREL will be most effective when there are many unpromising attributes that can be eliminated quickly, and when the promising attributes are all found along the same set of relational paths. In the next section we describe an application we developed to evaluate the performance of our algorithm, and to determine if it can successfully learn from massive relational data sets.

5 Predicting the Evolution of the Web

The World Wide Web has received much study in recent years. Researchers have studied ways to classify Web pages into categories (e.g., Slattery and Craven [2001]), search for high quality pages (e.g., Kleinberg [1998], Page *et al.* [1998]), model the way Web pages acquire links over time (e.g., Barabasi and Albert [2000], etc.) One commonality among much of this work is that analyzing the content of Web pages in isolation seldom produces the best results–the links between pages often contain critical information that must be taken into account. Unfortunately, as we have seen, state of the art systems for building complex relational models are in incapable of scaling to data sets the size of the Web.

In this section we describe an application of VFREL to mining a massive Web data set. The goal is to build a model that accurately predicts if a Web page's popularity will rise or fall in the future. Such models would be useful, for example, to help improve search engine results for new pages, and to help designers create pages that people will reference. We estimate the popularity change in a Web page by counting the number of pages that point to it in one crawl, and trying to predict if the page will be linked to by five or more additional pages, five or more fewer pages, or within five of the same number of pages in a future crawl. We take into account 47 intrinsic attributes of nearly two million Web pages. We also make use of relational information that includes seven object sources and millions of relations.

Our application begins with a crawl of approximately 1.7 million Web pages from .edu domains that was gathered in early June of 2001. The crawl contains pages from 31k unique Web hosts and uses 23 GB of disk space. It was gathered starting from a small set of seed Web pages (Google's top 20 results for the query 'university') and performing a breadth-first crawl until no more files would fit on the disk[3]. The crawl ran on a cluster of five 1 Ghz Linux machines, and took approximately 3 days to finish. We gathered a second crawl, using the same procedure and set of seed pages, in

[2]If the estimate is exceeded we report the global confidence that was actually achieved, or the algorithm can be run again with a better estimate if needed. Our experiments required just 13 iterations of the main loop.

[3]The version of Linux we used for these studies limited the number of files in a partition to 1.7 million. We plan on removing this limitation in a future study.

November of 2002. There were 563k pages that appeared in both crawls.

We put each of the pages that appeared in both crawls into a database (an object database which we implemented on top of GDBM). We used seven object sources to represent the domain, and their properties are as follows:

WebPage There are 563,083 Web page objects in our data set. Each has 47 attributes, including binary attributes to indicate the presence of the top 10 words according to information gain on the training set; the number of images; characterizations of alt text usage, script usage, color usage, etc.[4] and the PageRank [Page *et al.*, 1998] of the page within the subset of the Web covered by the first crawl.

WebPageLink There are 2,154,420 Web page link objects, one for each link between the pages in our data set. Each of these objects has a one-relation for its source and a one-relation for its destination.

Domain There are 21,069 domain objects in our data set. Each has a single categorical attribute, the Carnegie Classification (a publicly available classification of universities by their types) of the school it belongs to.

WebPageDomainLink There are 563,083 links from Web pages to their domain, one for each Web page. Each link has one numeric attribute, the depth of the page in the domain. Each also has a one-relation for the page and a one-relation for the domain.

Site We identified 412 sites in our crawl. A site is distinct from a domain by being managed by a small group of people and being about a well defined topic. We used a set of handcrafted regular expressions that examined URLs and identified sites including home pages, course pages, group pages, and project pages. The very low number of sites identified by our heuristics is problematic, and in future work we hope to improve this. Each site has an attribute that specifies its type.

WebPageSiteLink There were 1411 links between Web pages and sites. Each contains a one-relation to the page and a one-relation to the site.

SiteDomainLink There were 412 links from sites to their domains. Each has a single attribute, the depth of the site in the domain. Each also has a one-relation for the site and a one-relation for the domain.

Note that conceptually this domain could be modeled without the Link objects. We modeled it this way for several reasons: it is the best way to encode the many-many relation between WebPage objects in our database; it is conceptually simpler to have all links modeled the same way; it is cleaner and more extensible as we add additional features to the links (which we plan to do in future work); and it does not hurt efficiency compared to the other method.

We built index structures so that any relation could be followed by accessing the index on disk, and then loading the related objects from the GDBM on-disk hash table that contains them. The resulting database and associated index structures took on order of a day to construct on a 1Ghz PIII, and occupy approximately 900MB of disk space. Reading all the objects from disk in random order takes about 450 seconds. Notice that many of the attributes in our domain are numeric. We turn these attributes into categorical ones as needed by dividing the attribute into ten approximately equal-frequency regions. Each WebPage object has a target attribute, whose value is 'Gain5' if the number of links to the page in the new crawl is at least 5 greater than in the original crawl, 'Lose5' if the number of links to the page in the new crawl is at least 5 less than in the original crawl, and 'Same' otherwise. We evaluated the learning algorithms in this domain by removing the target attribute from a randomly selected 30% of the WebPage objects, using the learning algorithms to build models on the data set, and using the models to fill in these missing labels.

For these experiments we set VFREL's parameters as follows: maximum depth, $d = 5$; global confidence, $\delta^* = 5\%$; N, number of features to select $= 100$; and StepSize began at 1,000 and was doubled in every iteration where feature selection did not shrink the size of the traversal tree. We used the C4.5 decision tree learner [Quinlan, 1993] as the propositional learner. We selected this learner over a scalable propositional learner for two reasons: the N-attribute flattened training examples for the 563k Web page objects fit in RAM, and we wanted to make the contribution of our relational feature selection algorithm easier to evaluate. In future work we plan on combining VFREL with the scalable VFDT decision tree induction algorithm [Domingos and Hulten, 2000]. We ran our system in parallel on a cluster of five 1Ghz Pentium III workstations running Linux with RAM sizes ranging from 256MB to 512MB.

We compared our system to simply flattening the database and passing the flattened data to C4.5. With our computing resources we were able to flatten depths up to 2, and the flattened data sets are Flat0 (no relational attributes), Flat1, and Flat2 below. We also compared to one of the leading models of Web evolution, the preferential attachment model [Barabási *et al.*, 2000]. The preferential attachment model proposes that links are constantly being added to the Web, and that the probability that any particular page is the target of the next link is proportional to the number of links that it already has. We could not estimate the parameters needed to apply this model directly in our setting. Instead, we used the insight it is based on and built a decision tree on a single relational attribute: the number of pages that point to the target page (non-discretized).

We ran VFREL and Flat0-2 with all of their attributes (-full below) and also after performing additional feature selection to select the best 20 attributes in each setting. Table 3 contains the results of our experiments. Using 20 attributes yielded the best results for every system. VFREL with its 20 best attributes achieved the highest accuracy of any of the algorithms we considered. Note that while the differences in error rate are small on a percentage basis, they were measured on 169k test objects and represent real differences in performance. Also notice that increasing the depth of attributes

[4]Many of these attributes were gathered with the WebSAT toolkit: http://zing.ncsl.nist.gov/WebTools/

Table 3: Results of the comparison between VFREL, flattening depth 0 - 2, the preferential attachment model, and predicting the most common class, MCC, which was Same. We show VFREL and Flat0 - 2 with their full feature set and after doing additional feature selection.

Algorithm	Test Set Error	# Nodes	# Attribs
MCC	10.2%	0	0
PrefAtt	8.2%	5	1
Flat0	10.9%	18,372	20
Flat1	8.5%	11,663	20
Flat2	8.2%	9,741	20
VFREL	8.1%	7,465	20
Flat0-full	11.2%	10,197	47
Flat1-full	8.8%	15,117	50
Flat2-full	8.3%	10,308	330
VFREL-full	8.6%	14,289	100

considered results in smaller, more accurate models in our experiments. This suggests that these deeper attributes actually do contain valuable information for our task, and that it may be beneficial to explore further than depth 5 – we plan on doing this in future work. The runtimes for generating the flattened data sets were (in CPU + system hours): Flat0, .27; Flat1, .30; Flat2, 12.9. We estimate from generating the first 10k examples that Flat4 would have taken 54 days, and we estimate from generating the first 1,000 examples that Flat5 would have taken 261 days. Our system was able to generate values for the best 100 features up to depth 5 in 20 days of CPU time (4 days of wall time because it ran in parallel). VFREL is thus an order of magnitude faster than directly flattening the data, and produces the most accurate model of any of the systems we evaluated.

At the beginning of its run, VFREL was forced to follow 56 relational paths from each Web page to gather the objects needed to calculate the 3,536 attributes in $ATT(WebPage, 5)$ (after the obviously redundant ones were removed). By the end of the run it was following just 14 paths for each Web page. Every selected relational path begins by following the 'linked from' relation from the target object (that is, all selected relational attributes are properties of pages that point to the target page). After that, the 'links to', 'linked from', and 'domain' relations were used. None of the Site related attributes or relations were used to calculate the 100 best attribute values. We believe this will change when we improve our site identification heuristics.

The top 20 attributes included attributes formed using every aggregation we allowed except for mode. Eleven of them were aggregations of the PageRank of pages that pointed to the target, or were linked (in either direction) to pages that pointed to the target. Other selected features included aggregations of counts of Web pages, of depths of pages in their domain, of the number of words in link anchors, and of the size of related pages in bytes. The best attribute was the preferential attachment one, the count of the number of pages that point to the target. By examining the decision tree produced by C4.5 we determined that the information in the PageRank attributes was mostly captured by the preferential attachment attribute. We found the attributes that contributed to our system's improvement over the preferential attachment model were properties of other pages pointed to by the pages that pointed to the target, like the variance of the domain depths of the other pages pointed to by pages pointing to the target, and the popularity (as measured by the number of inlinks) of the other pages they point to. These features are a depth of 5 from the target class, and it is unlikely that they would have been found by other relational learning systems. We believe that properties of the pages pointing to the target are important for this prediction task because people find the target page (a prerequisite to linking to the page) through these links.

Generating attribute values for the median hundred Web pages out of the first thousand (before any feature selection) took 3,488 seconds and required that nearly 5 million object be loaded from disk. In the last iteration of VFREL's main loop, when it was exploring just 14 relational paths, the median 100 objects took just 153 seconds and 420k object loads – an improvement of an order of magnitude by either metric. There was a great deal of variance in the time it took to generate attributes for 100 objects. In fact, some single Web pages, even on the final iteration with only 14 relational paths, required thousands of seconds and millions of object loads. We examined some of these Web pages and found them to be extremely highly connected (tens of thousands of in links), on very large domains (with many tens of thousands of pages), or both. In future work we plan on exploring the use of online aggregations [Hellerstein *et al.*, 1997] to reduce the time needed to generate attribute values for these highly connected objects.

6 Related Work

Learning from relational data has been extensively studied by the inductive logic programming (ILP) community [Lavrac and Dzeroski, 1994]. In general, ILP learns models from a richer class than our work (for example, learning recursive concepts), but is also generally believed to be very inefficient for large databases. Blockeel *et al.* [1999] developed a scalable ILP system named TILDE that effectively flattens relational data into what they call interpretations and then uses a version of FOIL [Quinlan, 1990], modified to make efficient access to data from disk, on these interpretations. TILDE was evaluated on a synthetic data set with 100,000 training examples. VFREL scales to much larger data sets by using sampling to focus on relevant attributes and relations. Slattery and Craven [2001] extensively studied the use of relational learning for hyper-text documents. They developed a hybrid algorithm that combines Naive Bayes, FOIL, and many insights into the nature of the Web, and applied it to several Web mining tasks. Their focus, however, was not on scaling, and the largest data set they considered contained on the order of thousands of Web pages, while ours contains millions.

7 Future Work

Directions for future work on VFREL include: more closely integrating it with a scalable propositional learning algorithm

(for example VFDT); modifying learners to exploit information from the data generation process (for example, when a relation is missing, many related attributes simultaneously have missing values); extending it to the case where the contents of object sources change over time; modifying it to iterate between feature selection and learning phases; and applying it to other domains.

Future directions for our Web application include: performing a similar study with a larger Web crawl; performing a similar study on a more volatile portion of the Web (perhaps .com); adding more intrinsic attributes to the objects (words on links, more text, etc.); building models to predict which links will appear over time; and building models from the stream of pages that a crawler finds as it finds them.

8 Summary

In this paper we explored some of the issues that prevent relational learning algorithms from scaling to very large data sets. We developed a system, VFREL, which uses efficient data access and sampling to efficiently explore the space of relational attributes. We used VFREL to mine data sets containing millions of objects and links, and found it to build models that were more accurate than those produced by any of the systems we evaluated, discover novel relational attributes, and work an order of magnitude faster than the alternative approaches.

Acknowledgments

This research was partly supported by an NSF CAREER grant to the second author, by ONR grant no. N00014-02-1-0408, and by a gift from the Ford Motor Co.

References

[Barabási *et al.*, 2000] A. L. Barabási, R. Albert, and H. Jong. Scale-free characteristics of random networks: The topology of the World Wide Web. *Physica A*, 281:69–77, 2000.

[Blockeel *et al.*, 1999] H. Blockeel, L. D. Raedt, and N. Jacobs. Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery*, 3(1):59–93, 1999.

[Domingos and Hulten, 2000] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, Boston, MA, 2000. ACM Press.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1300–1307, Stockholm, Sweden, 1999. Morgan Kaufmann.

[Gehrke *et al.*, 1999] J. Gehrke, V. Ganti, R. Ramakrishnan, and W.-L. Loh. BOAT: optimistic decision tree construction. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 169–180, Philadelphia, PA, 1999. ACM Press.

[Getoor *et al.*, 2001] L. Getoor, N. Friedman, D. Koller, and B. Tasker. Learning probabilistic models of relational structure. In *Proceedings of the 18th Intern. Conference on Machine Learning*, pages 170–177, San Francisco, CA, 2001. Morgan Kaufmann.

[Hellerstein *et al.*, 1997] J. Hellerstein, P. Hass, and H. Wang. Online aggregation. In *Proceedings of the SIGMOD Intern. Conferencce on Managment of Data*, Tuscon, AZ, 1997.

[Hoeffding, 1963] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[Hulten and Domingos, 2002] G. Hulten and P. Domingos. Mining complex models from arbitrarily large databases in constant time. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 525–531, Edmonton, Canada, 2002. ACM Press.

[Jensen and Neville, 2002a] D. Jensen and J. Neville. Autocorrelation and linkage cause bias in evaluation of relational learners. In *Proceedings of the Twelfth International Conference on Inductive Logic Programming*, Sydney, Australia, 2002. Springer.

[Jensen and Neville, 2002b] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 259–266, Sydney, Australia, 2002. Morgan Kaufmann.

[Jensen and Neville, 2002c] D. Jensen and J. Neville. Schemas and models. In *Proceedings of the Multi-Relational Data Mining Workshop, 8th SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining*, Edmonton, Canada, 2002. ACM Press.

[Jensen, 1998] D. Jensen. Statistical challenges to inductive inference in linked data. In *Preliminary papers of the 7th Intern. Workshop on Artificial Intelligence and Statistics*, 1998.

[Kleinberg, 1998] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, Baltimore, MD, 1998. ACM Press.

[Kohavi and John, 1997] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2), 1997.

[Kononenko, 1994] I. Kononenko. Estimating attributes: Analysys and extensions of relief. In F. Bergadano and L. D. Raedt, editors, *Proceedings of the European Conference on Machine Learning*, 1994.

[Lavrac and Dzeroski, 1994] N. Lavrac and S. Dzeroski. *Inductive logic programming: techniques and applications*. Chichester, UK: Ellis Horwood, 1994.

[Moore and Lee, 1997] A. W. Moore and M. S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, 1997.

[Page *et al.*, 1998] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, Stanford, CA, 1998.

[Quinlan, 1990] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Quinlan, 1993] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Shafer *et al.*, 1996] J. C. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of the Twenty-Second International Conference on Very Large Databases*, pages 544–555, Bombay, India, 1996. Morgan Kaufmann.

[Slattery and Craven, 2001] S. Slattery and M. Craven. Relational learning with statistical predicate invention: better models for hypertext. *Machine Learning*, 43(1/2), 2001.

Representational Power of Probabilistic-Logical Models: From Upgrading to Downgrading*

Kristian Kersting
Institute for Computer Science
Machine Learning Lab
Albert-Ludwigs University of Freiburg
Georges-Koehler-Allee, Building 079
79110 Freiburg, Germany

1 Position Statement

There is a diversity of *probabilistic-logical models* (PLM). No clear understanding of the relative advantages and limitations of different formalisms and their language concepts has yet emerged. To overcome this, we propose to *downgrade* highly expressive PLMs. This method has several advantages: one can profit from existing research on PLMs and inherit unique semantics, and inference and learning algorithms. Moreover, there is a clear relationship between the new PLM and its more expressive counterpart. No single existing approach is devalued.

2 Motivation

In recent years, there has been an increasing interest in probabilistic-logical models (PLMs). PLMs integrate probability theory with some first order logic. Traditionally, a probabilistic formalism like Bayesian networks or hidden Markov models is selected and *upgraded* by incorporating some logic such as entity-relationship (ER) models or Prolog. Real-world data applications have shown the potential of PLMs e.g. in query optimization [Getoor *et al.*, 2001], computational biology [Segal *et al.*, 2001; Kersting *et al.*, 2003] and web mining [Anderson *et al.*, 2002].

Despite these successes, the field of (learning) PLMs is quite complex and confusing. PLMs *"have been developed in several related, but different, subareas of artificial intelligence (reasoning under uncertainty, inductive logic programming, machine learning, and knowledge discovery and data mining)"* as stated by Lise Getoor and David Jensen in SRL-2003's CFP. Each subarea focuses on its own language concepts. Consider Table 1 which lists a subset of proposed formalisms [1]. The language concepts vary from acyclic to cyclic models, from logically structured dependencies among random variables to states, from finite to continuous random variables, and from functor-free languages to Prolog. They each have their respective merits. However, the *upgrading* mentality together with concentrating on particular language concepts makes a general understanding of PLMs and learning PLMs difficult – if not impossible.

*This is a position statement for the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data

[1] Avi Pfeffers's interesting PhD thesis provides some more references, [Pfeffer, 2000].

3 Downgrading

Downgrading consists of two steps.

(Step 1) Choose a generally applicable (learning) PLM.

The PLM should cover the basic language concepts proposed in the different scientific subareas:

- **Relations** among random variables or states to model uncertainty. This subsumes interesting concepts such as referential and existential uncertainty.
- **Functors** allow to consistently encode temporal correlations (dynamic Bayesian networks), complex long distance correlations (stochastic grammars), *named by structure* entities as they are common in semi-structured data (e.g. XML), and general data structures (lists, trees, etc). Functors incorporate flexible memory capabilities.
- **Finite**, **discrete** and **continuous random variables** together provide compact models which are applicable in a broad field of applications such as classification, clustering, and regression.
- Often, e.g. in computational biology, one is interested not only to simulate but to gain insight into, and understand the underlying processes. Therefore, PLMs should be **interpretable**.
- Learning the PLM should facilitate to define and to specify both deterministic and probabilistic **background knowledge**. This not only makes it possible to specify the huge amount of expert knowledge often available but also to break complex questions into subtasks still taking care of dependencies among the subtasks.

It is likely that the very general PLM is prohibitively powerful for a problem at hand. Therefore,

(Step 2) downgrade it to strike the right balance between expressivity and learnability.

Compared to upgrading, downgrading has the following advantages. First, the downgraded PLM inherits unique semantics, and inference and learning algorithms. Second, downgrading does not focus on a particular PLM. Instead it systematically investigates the impact of language concepts. A general understanding of PLMs and learning PLMs is likely to emerge.

PLM	Probabilistic Formalism	Logic
Probabilistic Horn Abduction (PHA) [Poole, 1993]	Bayesian Networks	Prolog
PRISM [Sato, 1995]	Stochastic Grammars	Prolog
Stochastic Logic Programs (SLPs) [Muggleton, 1996; Cussens, 2000]	Stochastic Grammars	Prolog
Probabilistic Logic Programs (PLPs) [Ngo and Haddawy, 1997]	Bayesian Networks	Prolog
Bayesian Logic Programs (BLPs) [Kersting and De Raedt, 2001]	Bayesian Netwoks	Prolog
Relational Baysian networks (RBNs) [Jaeger, 1997]	Bayesian Networks	Relational
Probabilistic Relational Models (PRMs) [Friedman *et al.*, 1999]	Bayesian Networks	ER Models
Relational Markov Models (RMMs) [Anderson *et al.*, 2002]	Markov Models	Relational
Logical Hidden Markov Models (LOHMMs) [Kersting *et al.*, 2002]	Hidden Markov Models	Iterative Clauses

Table 1: A collection of probabilistic-logical models together with their underlying probabilistic and logical formalism.

Initial attempts of downgrading have been done. Restricting SLPs to iterative clauses leads in principle to LOHMMS [Kersting *et al.*, 2002]. [Sato and Kameya, 2001] propose an EM algorithm for parameter estimation of PRISMs showing that the algorithm exhibits the same complexity for hidden Markov models and stochastic context free grammars as the specialized counterparts.

4 Related Work

Downgrading is related to work comparing the expressivity of different PLMs [Kersting and De Raedt, 2001; Jensen and Neville, 2002]. Furthermore, downgrading is akin to contemporary considerations in the *inductive logic programming* and the *Baysian networks* communities. E.g. Kevin Murphy motivates the development of his Matlab *Baysian Network Toolbox* as follows: *"I was fed up with reading papers where all people do is figure out how to do exact inference and/or learning in a model which is just a trivial special case of a general Bayes net, e.g., input-output HMMs, coupled-HMMs, auto-regressive HMMs. My hope is that, by releasing general purpose software, the field can move on to more interesting questions"*, see `http://www.ai.mit.edu/~murphyk/Software/BNT/bnt.html`. For similar reasons, we initiated a repository for (learning) PLMs at `http://www.informatik.uni-freiburg.de/~kersting/plmr/`.

Acknowledgements

The position of the author was partially consolidated by the interesting discussion within the successful EU assessment project IST-2001-33053 (APRIL). The author would like to thank all collaborators and especially the reviewers. Special thanks to Lise Getoor and David Jensen for discussions about a PLM repository.

References

[Anderson *et al.*, 2002] C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov Models and their Application to Adaptive Web Navigation. In *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, 2002.

[Cussens, 2000] J. Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2000.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-1999)*, pages 1300–1307, 1999.

[Getoor *et al.*, 2001] L. Getoor, B. Taskar, and D. Koller. Selectivity estimation using probabilistic models. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2001)*., 2001.

[Jaeger, 1997] M. Jaeger. Relational Bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 266–273. Morgan Kaufmann, 1997.

[Jensen and Neville, 2002] D. Jensen and J. Neville. Schemas and models. In *Proceedings of the Multi-Relational Data Mining Workshop, 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[Kersting and De Raedt, 2001] K. Kersting and L. De Raedt. Towards Combining Inductive Logic Programming with Bayesian Networks. In *Proceedings of the 11th International Conference on Inductive Logic Programming*, volume 2157 of *LNAI*, pages 118–131. Springer, 2001.

[Kersting *et al.*, 2002] K. Kersting, T. Raiko, and L. De Raedt. Logical Hidden Markov Models (Extended Abstract). In *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM-02)*, Spain, November 2002.

[Kersting *et al.*, 2003] K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards Discovering Structural Signatures of Protein Folds based on Logical Hidden Markov Models. In *Proceedings of the Pacific Symposium on Biocomputing*, 2003.

[Muggleton, 1996] S. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.

[Ngo and Haddawy, 1997] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.

[Pfeffer, 2000] A. J Pfeffer. *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford University, 2000.

[Poole, 1993] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.

[Sato and Kameya, 2001] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[Sato, 1995] T. Sato. A Statistical Learning Method for Logic Programs with Distribution Semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP-1995)*, pages pp. 715 – 729, 1995.

[Segal *et al.*, 2001] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. In *Proceedings of the 9th International Conference on Intelligent Systems For Molecular Biology (ISMB)*, 2001.

Logical Markov Decision Programs

Kristian Kersting and **Luc De Raedt**
Machine Learning Lab,
University of Freiburg,
Georges-Koehler-Allee, Building 079,
79110 Freiburg, Germany

Abstract

Motivated by the interest in relational reinforcement learning, we introduce a novel representation formalism, called logical Markov decision programs (LOMDPs), that integrates Markov Decision Processes with Logic Programs. Using LOMDPs one can compactly and declaratively represent complex relational Markov decision processes. Within this framework we then develop a theory of reinforcement learning in which abstraction (of states and actions) plays a major role. The framework presented should provide a basis for further developments in relational reinforcement learning.

1 Introduction

In the past few years, there has been a lot of work on extending probabilistic and stochastic frameworks with abilities to handle objects, see e.g. [Anderson *et al.*, 2002; Džeroski *et al.*, 2001; Friedman *et al.*, 1999; Kersting and De Raedt, 2001; Kersting *et al.*, 2003; Muggleton, 1996]. From an inductive logic programming or relational learning point of view, these approaches are upgrades of propositional representations towards the use of relational or computational logic representations. Various successes in this direction have been reported. Indeed, [Friedman *et al.*, 1999] and [Kersting and De Raedt, 2001] upgrade Bayesian networks, [Muggleton, 1996] upgrades stochastic context free grammars, [Anderson *et al.*, 2002] and [Kersting *et al.*, 2003] upgrade (hidden) Markov models.

The first contribution of this paper is the introduction of a novel representation formalism, called *logical Markov decision programs* (LOMDPs), that combines *Markov decision processes* with computational logic. The result is a flexible and expressive framework for defining MDPs that are able to handle structured objects as well as relations and functors. For MDPs, such a framework grounded in computational logic, was still missing. Only [Boutilier *et al.*, 2001] report on combining MDPs with Reiter's situation calculus. However, as we argue in Section 7, it is more complex and model-free reinforcement learning techniques have yet not been addressed within this framework. LOMDPs share - with the other upgrades of propositional representations - two advantages. First, logical expressions (in the form of clauses, rules or transitions) may contain variables and as such make *abstraction* of many specific *grounded* rules or transitions. This allows one to compactly represent complex domains. Secondly, because of this abstraction, the number of parameters (such as rewards and probabilities) in the model is significantly reduced. This in turn allows - in principle - to speed up and simplify the learning because one can learn at the *abstract* level rather than at the *ground* level.

Many fascinating machine learning techniques have been developed under the name reinforcement learning (RL) in the context of MDPs over the last few decades, cf. [Sutton and Barto, 1998]. Recently, there has also been an increased attention for dealing with relational representations and objects in reinforcement learning, see e.g. [Džeroski *et al.*, 2001; Finney *et al.*, 2002] Many of these works have taken a practical perspective and have developed systems and experiments that operate in relational worlds. At the heart of these systems there is often a function approximator (often a logical decision tree) that is able to assign values to sets of states and to sets of state–action pairs. So far, however, a theory that explains why this approach works seems to be lacking. The second and most important contribution of this paper is a first step into the direction of such a theory. The theory is based on a notion of abstract states and abstract policies represented by logical expressions. An abstract state represents a set of concrete states and an abstract policy is then a function from abstract states to actions. All ground states represented by the same abstract state are essentially assigned the same action. This is akin to what happens with (relational) reinforcement learning using (logical) decision trees [Džeroski *et al.*, 2001], where each leaf of the decision tree represents an abstract state and where states classified in the same leaf obtain the same value or action. Within the LOMDP framework abstract policies can easily be represented. The abstract value function (assigning values to abstract states or state action pairs) is defined as the average values of the states or state action pairs they represent. We will show that these abstract value functions cannot in general be learned using traditional MDP methods. This in turn provides some new insights into relational reinforcement learning approaches.

We proceed as follows. After introducing some mathematical preliminaries in Section 2, we present the LOMDP framework in Section 3. Section 4 defines abstract policies and shows how to compute the value of an abstract policy. This

results in the LQ learning algorithm presented in Section 5. The algorithm is experimentally evaluated in Section 6. Before concluding, we discuss related work.

2 Preliminaries

As logic programs and Markov decision processes will be used throughout this paper as the underlying mathematical concepts, we now briefly introduce their key concepts.

2.1 Logic

A *first-order alphabet* Σ is a set of relation symbols $\mathtt{r}$ with arity $m \geq 0$, and a set of functor symbols $\mathtt{f}$ with arity $n \geq 0$. If $n = 0$ then $\mathtt{f}$ is called a constant, if $m = 0$ then p is called a proposition. An *atom* $\mathtt{r}(\mathtt{t}_1, \ldots, \mathtt{t}_m)$ is a relation symbol $\mathtt{r}$ followed by a bracketed n-tuple of terms $\mathtt{t}_i$. A *term* is a variable $\mathtt{V}$ or a functor symbol immediately followed by a bracketed n-tuple of terms $\mathtt{t}_i$, i.e., $\mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$. A conjunction is a set of atoms. A conjunction A is said to be θ-subsumed by a conjunction B, denoted by $A \leq_\theta B$, if there exists a substitution θ such that $B\theta \subset A$. A term, atom or clause E is called *ground* when it contains no variables. The *most general unifier* (MGU) for atoms a and b is denoted by $\mathrm{mgu}(a, b)$. The *Herbrand base* of Σ, denoted as hb_Σ, is the set of all ground atoms constructed with the predicate and functor symbols in the alphabet Σ.

2.2 Notation

Atoms are written in lower case a, set of of atoms in upper case A, and sets of sets of atoms in bold, upper case $\mathbf{A}$. To highlight that a (resp. A and $\mathbf{A}$) may not be ground (i.e. it may contain variables), we will write $\mathbb{a}$ (resp. $\mathbb{A}$ and $\mathbb{A}$).

2.3 Markov Decision Processes

A Markov decision process (MDP) is a tuple $\mathbf{M} = (S, A, \mathbf{T}, \lambda)$. To avoid ambiguities, we will sometimes index the elements by $\mathbf{M}$. Here, S is a set of system states, i.e. propositions. The agent has available a finite set of actions $A(z) \subseteq A$ for each state $z \in S$ which cause stochastic state transitions. For each $z, z' \in S$ and $a \in A(z)$ there is a transition T in $\mathbf{T}$ which is an expression of the form $z' \xleftarrow{p:r:a} z$. The transition denotes that with probability $P(z, a, z') := p$ action a causes a transition to state z' when executed in state z. We have for each $z \in S$ and $a \in A(z)$ that $\sum_{z' \in S} P(z, a, z') = 1$. For a transition the agent gains an expected next reward $R(z, a, z') := r$. In case that the reward function R is probabilistic (mean value depends on the current state and action only) the MDP is called *nondeterministic*, otherwise *deterministic*. In this paper, we only consider MDPs with stationary transition probabilities and stationary, bounded rewards.

A (stationary) deterministic policy $\pi : S \mapsto A$ is a set of expressions of the form $a \leftarrow z$ for each $z \in S$ where $a \in A(s)$. It denotes a particular course of actions to be adopted by an agent, with $\pi(z) := a$ being the action to be executed whenever the agent finds itself in state z. We assume an infinite horizon and also that the agent accumulates the rewards associated with the states it enters. To compare policies, we use the expected total discounted reward as our optimality criterion, i.e., future rewards are discounted by $0 \leq \lambda < 1$. The value of a policy π can be shown to be $V_\pi(z) = \sum_{z' \xleftarrow{p:r:a} z \in \mathbf{T}} p \cdot [r + \lambda \cdot V_\pi(z')]$. The value of π at any initial state z can be computed by solving this system of linear equations. A policy π is optimal if $V_\pi(z) \geq V_{\pi'}(z)$ for all $z \in S$ and policies π'. A (stationary) nondeterministic policy π maps a state to a distribution over actions. The value of π is then the expectation according to this distribution.

3 Logical Markov Decision Programs

The *logical component* of a MDP corresponds to a *finite state automaton*. This is essentially a propositional representation because the state and action symbols are flat, i.e. not structured. The key idea underlying *logical Markov decision programs* (LOMDPs) is to replace these flat symbols by abstract symbols.

Definition 1. *An abstract state is a conjunction $\mathbb{Z}$ of logical atoms, i.e., a logical query. In case of an empty conjuction, we write $\emptyset$.*

Abstract states represent sets of states. More formally, we have that a state Z is a (finite) conjunction of ground facts over the alphabet Σ, i.e. a logical interpretation, a subset of the Herbrand base. In the blocks world, one possible state Z is $\mathtt{on(a,b)}$, $\mathtt{on(b,fl)}, \mathtt{bl(a)}, \mathtt{bl(b)}, \mathtt{cl(a)}, \mathtt{cl(fl)}$ where $\mathtt{on(a,b)}$ denotes that object $\mathtt{a}$ is on $\mathtt{b}$, $\mathtt{cl(a)}$ states that $\mathtt{a}$ is clear, $\mathtt{bl(a)}$ denotes that $\mathtt{a}$ is a block, and $\mathtt{fl}$ refers to the floor. An abstract state $\mathbb{Z}$ is e.g. $\mathtt{on(X,Y)}$, $\mathtt{bl(Y)}, \mathtt{bl(X)}$. It represents all states (over the given alphabet Σ) that have two blocks on one another. Formally, speaking, we have that an abstract state $\mathbb{Z}$ represents all states Z for which there exists a substitution θ such that $\mathbb{Z}\theta \subseteq Z$. Let $S(\mathbb{Z})$ denote this set of states. The substitution in the previous example is $\{\mathtt{X/a}, \mathtt{Y/b}\}$. By now we are able to define abstract transitions.

Definition 2. *An* abstract transition T *is an expression of the form* $\mathbb{H} \xleftarrow{p:r:\mathbb{a}} \mathbb{B}$ *where* $\mathrm{P(T)} := p \in [0, 1]$, $\mathrm{R(T)} := r \in [0, 1]$, $\mathbb{a}$ *is an abstract action, and* $\mathrm{body(T)} := \mathbb{B}$ *and* $\mathrm{head(T)} := \mathbb{H}$ *are abstract states.*

We assume T to be range-restricted, i.e., $\mathrm{vars}(\mathbb{H}) \subseteq \mathrm{vars}(\mathbb{B})$, and $\mathrm{vars}(\mathbb{a}) \subseteq \mathrm{vars}(\mathbb{B})$, so that an abstract transition relies on the information encoded in the current state only. The semantics of an abstract transition[1] are:

> *If the agent is in a state Z, such that $\mathbb{B} \leq_\theta Z$, then it will go to the state $Z' := [Z \setminus \mathbb{B}\theta] \cup \mathbb{H}\theta$ with probability p when performing action $\mathbb{a}\theta$ receiving an expected next reward of r.*

For illustration purposes [2], consider the following abstract transition, which moves block $\mathtt{X}$ from $\mathtt{Y}$ to the floor with probability 0.9:

$$\mathtt{on(X,fl)}, \mathtt{cl(X)cl(Y)} \xleftarrow{0.9:-1:\mathtt{mv\_fl(X)}} \mathtt{on(X,Y)}, \mathtt{cl(X)}$$

[1] We implicitly assume that an abstract action has some preconditions

[2] Please note that we employ functor-free examples throughout the paper for the sake of simplicity. Abstract states $\mathbb{Z}$, actions $\mathbb{A}$, and transitions $\mathbb{T}$ can include functors. All proofs remain valid.

Applied to state *Exp*

$$\texttt{on}(\texttt{a},\texttt{b}),\texttt{on}(\texttt{b},\texttt{fl}),\texttt{on}(\texttt{c},\texttt{fl}),$$
$$\texttt{cl}(\texttt{a}),\texttt{cl}(\texttt{c}),\texttt{bl}(\texttt{a}),\texttt{bl}(\texttt{b}),\texttt{bl}(\texttt{c})$$

the abstract transition tells us that when we execute $\texttt{mv\_fl}(\texttt{a})$ the successor state will be

$$\texttt{on}(\texttt{a},\texttt{fl}),\texttt{on}(\texttt{b},\texttt{fl}),\texttt{on}(\texttt{c},\texttt{fl}),$$
$$\texttt{cl}(\texttt{a}),\texttt{cl}(\texttt{b}),\texttt{cl}(\texttt{c}),\texttt{bl}(\texttt{a}),\texttt{bl}(\texttt{b}),\texttt{bl}(\texttt{c})$$

with probability 0.9 gaining a reward of -1. One can see that this implements a kind of first-order variant of probabilistic STRIPS operator, cf. [Hanks and McDermott, 1994].

As LOMDPs typically consist of a set $\mathbb{T}$ of multiple abstract transitions there are two constraints to be imposed in order to obtain meaningful LOMDPs. First, let $\mathbb{B}$ be the set of all bodies of abstract state transitions in the LOMDP (modulo variable renaming). For $\mathbb{B} \in \mathbb{B}$, let $\mathbb{A}(\mathbb{B})$ denote the set of all abstract actions $\mathbb{a}$ such that $\mathbb{H} \xleftarrow{p:r:\mathbb{a}} \mathbb{B}$ is in the LOMDP. We require

$$\forall \mathbb{B} \in \mathbb{B}, \forall \mathbb{a} \in \mathbb{A}(\mathbb{B}) \sum_{\substack{\mathrm{T}\in\mathbb{T},\\ \mathrm{body(T)}=\mathbb{B},\\ \mathrm{act(T)}=\mathbb{a}}} \mathrm{P(T)} = 1.0. \qquad (1)$$

This condition guarantees that all abstract successor states are specified when executing an abstract action in an abstract state and that their probabilities sum to 1. Secondly, we need a way to cope with contradicting transitions and rewards. Indeed, consider the two transitions $\texttt{e} \xleftarrow{1:-1:\texttt{a}} \texttt{d}$ and $\texttt{g} \xleftarrow{1:-2:\texttt{a}} \texttt{f}$, and state $Z = \{\texttt{d},\texttt{f}\}$. The problem with these transitions is that the first transition says that if we execute $\texttt{a}$ in Z we will go with probability 1 to state $Z' = \{\texttt{e},\texttt{f}\}$ whereas the second assigns a probability of 1 to state $Z'' = \{\texttt{d},\texttt{g}\}$. There are essentially two ways to deal with this situation. On the one hand, one might want to combine the two transitions and assign a probability of 0.5 to both Z' and Z'' for Z. On the other hand, one might want to have only one of rule of fire. In this paper, we take the second approach because this allows us to consider the transitions more independently of one another. This in turn will simplify learning and yields locally interpretable models. We assume a total order $\prec$ over all action-body pairs in $\mathbb{T}$ and do a forward search among the pairs stopping with the first matching body such as in Prolog[3]. From now on, we assume $\mathbb{B}$ to be ordered w.r.t. $\prec$. We will give an example after the next definition.

By now we are able to formally define logical Markov decision programs.

Definition 3. *A logical Markov decision process (LOMDP) is a tuple* $\mathbb{M} = (\Sigma, \mathbb{A}, \mathbb{T}, \lambda)$ *where* Σ *is a logical alphabet,* $\mathbb{A}$*, is a set of abstract actions,* $\mathbb{T}$ *is a finite set of abstract state transitions based on actions in* $\mathbb{A}$*, and* $0 \leq \lambda < 1$ *is a discount factor, such that* (1) *holds.*

[3]We chose a total order for the sake of simplicity. A partial order $\prec$ among the pairs s.t. the set of pairs is well-founded, i.e., every descending chain of elements w.r.t. $\prec$ is finite, actually suffices. Then, the conflict resolution strategy is to select only those abstract transitions whose action-body pair is minimal. An example is given in [Kersting *et al.*, 2003] where a kind of subsumption (or generality) relation among $\mathbb{B}$ is employed. All theorems can be adapted accordingly.

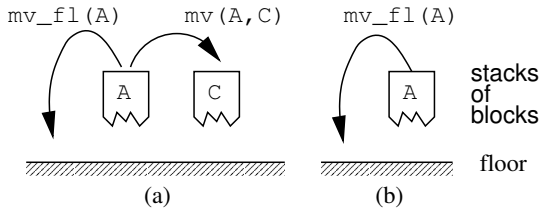


Figure 1: The two underlying patterns of the blocks world. Figure (a) shows the situation that there are at least two stacks of height > 0. Figure (b) shows the situation that there is only one stack left. The serrated cuts indicate that `A` (resp. `C`) can be on top of some other block or on the floor.

Before giving the semantics of LOMDPs, let us also illustrate LOMDPs on the *stack* example from the blocks world:

1:	absorb	$\xleftarrow{1.0:0.0:\texttt{absorb}}$	absorb.
2:	$\texttt{on}(\texttt{A},\texttt{fl}),\texttt{cl}(\texttt{A}),$ $\texttt{on}(\texttt{C},\texttt{D}),\texttt{cl}(\texttt{C}),$ $\texttt{cl}(\texttt{B})$	$\xleftarrow{0.9:-1:\texttt{mv\_fl(A)}}$	$\texttt{on}(\texttt{A},\texttt{B}),\texttt{cl}(\texttt{A}),$ $\texttt{on}(\texttt{C},\texttt{D}),\texttt{cl}(\texttt{C}).$
3:	$\texttt{on}(\texttt{A},\texttt{C}),\texttt{cl}(\texttt{A}),$ $\texttt{on}(\texttt{C},\texttt{D}),\texttt{cl}(\texttt{C}),$ $\texttt{cl}(\texttt{B})$	$\xleftarrow{0.9:-1:\texttt{mv(A,C)}}$	$\texttt{on}(\texttt{A},\texttt{B}),\texttt{cl}(\texttt{A}),$ $\texttt{on}(\texttt{C},\texttt{D}),\texttt{cl}(\texttt{C}).$
4:	absorb	$\xleftarrow{1.0:20:\texttt{stop}}$	$\texttt{on}(\texttt{A},\texttt{B}),\texttt{cl}(\texttt{A}),$ $\texttt{bl}(\texttt{B}).$

If the transition probabilities do not sum to 1.0 for an abstract action then there is an additional abstract transition for staying in the current abstract state. In order to understand the LOMDP *stack*, one has to understand the abstract states that govern the underlying patterns of the blocks world, cf. Figure 1. Two abstract states (the artificial `absorb` state excluded) together with the order in which they are presented cover all possible state action patterns because we can take advantage of symmetry in the blocks world. Transition 1 encodes the absorbing state. Transitions 2 and 3 cover the cases in which there are (at least) two stacks. Finally, transition 4 encodes the situation that there is only one stack, i.e. our goal state *stack*. Here, $\texttt{on}(\texttt{A},\texttt{B}),\texttt{cl}(\texttt{A}),\texttt{bl}(\texttt{B})$ are only used to describe the preconditions of $\texttt{mv}(\texttt{A},\texttt{B})$: the floor cannot be moved. When performing action $\texttt{mv}(\texttt{a},\texttt{b})$ in state *Exp* (see above) only abstract transitions 4 is firing. Similar, we can easily encode the *unstack* goal.

Note that we have not specified the number of blocks. The LOMDP represents all possible blocks worlds using only 6 abstract transitions, i.e. 12 probability and reward parameters, whereas the number of parameters of a propositional system explodes: for 4 blocks there are 73 states, for 7 blocks 37.663 states, and for 10 blocks $58.941.091$ states, resulting in an even higher number of transitions.

The semantics of LOMDPs are as follows.

Theorem 1. *Every LOMDP* $\mathbb{M} = (\Sigma, \mathbb{A}, \mathbb{T}, \lambda)$ *specifies a discrete MDP* $\mathbf{M}(\mathbb{M}) = (S, A, \mathbf{T}, \lambda)$.

Proof sketch: Let $\mathrm{hb}^{\mathrm{s}}_{\Sigma} \subset \mathrm{hb}_{\Sigma}$ be the set of all ground atoms built over abstract **s**tates predicates, and let $\mathrm{hb}^{a}_{\Sigma} \subset$

hb_Σ be the set of all ground atoms built over abstract **a**ction names. Now, construct $\mathbf{M}(\mathbb{M})$ from $\mathbb{M}$ as follows. The countable state set S consists of all finite subsets of hb^s_Σ. The set of actions $\mathbf{A}(Z)$ for state $Z \in S$ is given by $\mathbf{A}(Z) = \{\mathtt{a}\theta | \mathbb{H} \xleftarrow{p:r:\mathtt{a}} \mathbb{B} \in \mathbb{T}$ minimal (w.r.t. $\prec$) , $\mathbb{B} \leq_\theta Z\}$. We have that $|\mathbf{A}(Z)| < \infty$ holds. The probability $P(Z, a, Z')$ of a transition in $\mathbf{T}$ from Z to another state Z' after performing an action a is the probability value p associated to the unique abstract transition matching Z, a, and Z' normalized by the number of transitions of the form $Z'' \xleftarrow{a} Z$ in $\mathbb{T}$. If there is no abstract transition connecting Z and Z', the probability is zero. The bounded rewards are constructed in a similar way but are not normalized. □

From Theorem 1 and [Puterman, 1994, Theorem 6.2.5] it follows that:

Corollary 1. *For every LOMDP, there exists an optimal policy (for the ground states).*

Finally, LOMDPs generalize finite MDPs.

Proposition 1. *Every* finite *MDP is a propositional LOMDP in which all relation symbols have arity* 0*.*

4 Abstract Policies

Theorem 1 states that every LOMDP $\mathbb{M}$ specifies a discrete MDP $\mathbf{M}(\mathbb{M})$. Furthermore, Corollary 1 guarantees that there exists an optimal policy π for MDP $\mathbf{M}(\mathbb{M})$. Of course, this policy is extensional or propositional in the sense that it specifies for each ground state separately which action to execute. Specifying such policies for LOMDPs with large state spaces is cumbersome and learning them will require much effort. Therefore, we introduce *abstract policies* $\boldsymbol{\pi}$ which intentionally specify the action to take for an abstract state (or sets of states).

Definition 4. *An abstract policy* $\boldsymbol{\pi}$ *over* Σ *is a finite set of decision rules of the form* $\mathtt{a} \leftarrow \mathbb{L}$ *where* $\mathtt{a}$ *is an abstract action and* $\mathbb{L}$ *is an abstract state*[4].

The meaning of a decision rule $\mathtt{a} \leftarrow \mathbb{L}$ is that

> *if the agent is in a state Z such that $\mathbb{L} \leq_\theta Z$ then the agent will perform action $\mathtt{a}\theta$, denoted by $\boldsymbol{\pi}(Z)$.*

Usually, $\boldsymbol{\pi}$ consists of multiple decision rules. We apply the same conflict resolution technique as for abstract transitions, i.e. we use a total order $\prec$ among the decision rules. Let $\mathbb{L} = \{\mathbb{L}_1, \ldots, \mathbb{L}_m\}$ be the set of bodies in $\boldsymbol{\pi}$ (ordered w.r.t. $\prec$). We call $\mathbb{L}$ the *abstraction level* of $\boldsymbol{\pi}$. We assume that $\mathbb{L}$ covers all possible states of the LOMDP. This together with the total order guarantees that $\mathbb{L}$ forms a partition of the states. The equivalence classes $[\mathbb{L}_1], \ldots, [\mathbb{L}_m]$ induced by $\mathbb{L}$ are inductively defined by $[\mathbb{L}_1] = S(\mathbb{L}_1)$, and for $i \geq 2$, $[\mathbb{L}_i] = S(\mathbb{L}_i) \setminus \bigcup_{j=1}^{i-1} [\mathbb{L}_j]$. Because $\mathbb{L}$ generally does not coincide with $\mathbb{B}$ the following proposition holds.

Proposition 2. *Any abstract policy* $\boldsymbol{\pi}$ *specifies a* nondeterministic *policy* π *at the level of ground states.*

Let $\mathbb{M}$ be a LOMDP and let $\mathbf{M}(\mathbb{M})$ be the induced MDP. We define the expected reward of $\mathbb{L} \in \mathbb{L}$ to be the expected reward taken over all states in $[\mathbb{L}]$. Therefore, the expected discounted reward, if abstract policy $\boldsymbol{\pi}$ is used and the system is in abstract state $\mathbb{L}$, is defined to be

$$V_{\boldsymbol{\pi}}(\mathbb{L}) = \lim_{N \to \infty} E_{[\mathbb{L}]} \left[E_\pi \left\{ \sum_{k=1}^{N} \lambda^k r_{t+k} | Z_t = Z \right\} \right] \qquad (2)$$

where r_i denotes the value at time i of the reward received w.r.t. $\mathbf{M}(\mathbb{M})$ when following the ground level policy π induced by $\boldsymbol{\pi}$. The inner expectation E_π is conditioned on the system being in state $Z \in S$ at time t, denoted by $Z_t = Z$. The outer expectation $E_{[\mathbb{L}]}$ runs over all elements of $[\mathbb{L}]$. The series in (2) converges absolutely for the same reasons as for MDPs. Thus, the limit and the expectations are interchangeable in (2):

$$V_{\boldsymbol{\pi}}(\mathbb{L}) = E_{[\mathbb{L}]} \left[E_\pi \left\{ \sum_{k=1}^{\infty} \lambda^k r_{t+k} | Z_t = Z \right\} \right] . \qquad (3)$$

The abstract Q function is defined analogously. Now, an abstract policy $\boldsymbol{\pi}$ is *discount optimal at abstraction level* $\mathbb{L}$ for fixed λ whenever $V_{\boldsymbol{\pi}}(\mathbb{L}) \geq V_{\boldsymbol{\pi}'}(\mathbb{L})$ for all $\mathbb{L} \in \mathbb{L}$ and abstract policies $\boldsymbol{\pi}'$ at abstraction level $\mathbb{L}$. Note, that optimality at abstraction level $\mathbb{L}$ does not imply optimality at the level of ground states. This is because an abstract policy specifies the expected behaviour of a set of ground states. The problem is now to compute the value function $V_{\boldsymbol{\pi}}$.

Let $\mathbb{M} = (\Sigma, \mathbb{A}, \mathbb{T}, \lambda)$ be a LOMDP, and let $\boldsymbol{\pi}$ be an abstract policy at abstraction level $\mathbb{L} = \{\mathbb{L}_1, \ldots, \mathbb{L}_m\}$. Consider the finite MDP $\mathbf{L} = (\{l_1, \ldots, l_m\}, A_{\mathbf{L}}, \mathbf{T}_{\mathbf{L}}, \lambda)$ wich is constructed as follows.

Construction: Both $\mathbb{L}$ and $\mathbb{B}$ (the set of bodies in $\mathbb{T}$) induce partitions $\{[\mathbb{L}_1], \ldots, [\mathbb{L}_m]\}$ (resp. $\{[\mathbb{B}_1], \ldots, [\mathbb{B}_n]\}$) of $S_{\mathbf{M}(\mathbb{M})}$ because both are ordered. The state l_i corresponds to $[\mathbb{L}_i]$. Furthermore, all ground states belonging to $[\mathbb{L}_i] \cap [\mathbb{B}_k]$ have the same set of possible transitions. In other words, $[\mathbb{L}_i] \cap [\mathbb{B}_k]$ forms an equivalence class. Now, there is a transition $T \in \mathbf{T}_{\mathbf{L}}$ from state l_i to l_j when doing action a with probability

$$P(l_i, a, l_j) := \sum_{\mathbb{H} \xleftarrow{p:r:\mathtt{a}} \mathbb{B} \in \mathbb{T}} \mu([\mathbb{B}] | [\mathbb{L}_i]) \cdot p \cdot \mu([\mathbb{L}_j] | S(\mathbb{H}))$$

Here, $\mu(X|Y)$ is a probability function. The value $\mu(X|Y)$ for $X, Y \subset S_{\mathbf{M}(\mathbb{M})}$ is the probability that a randomly selected ground state in Y is an element of X. Because $\mathbf{M}(\mathbb{M})$ induces a unique probability distribution over all ground states, μ is uniquely specified. This follows from Theorem 1. Clearly,

$$\sum_{l_j} P(l_i, a, l_j) = 1 .$$

The intuition behind $P(l_i, a, l_j)$ is that it specifies $P(\mathbb{L}_i, \mathtt{a}, \mathbb{L}_j)$ for the corresponding abstract states. The probabilistic reward $R(l_i, a, l_j)$ depends only on l_i and A, and can

[4] We assume that $\mathtt{a}$ is applicable in $\mathbb{L}$.

be chosen [5] s.t. its mean value equals

$$R(l_i, a) := \sum_{l_j} P(l_i, a, l_j) \cdot R(l_i, a, l_j) \ . \quad \square$$

As the underlying MDP $\mathbf{M}(\mathbb{M})$ is not known, the problem specified by $\mathbf{L}$ appears to a learner to have a non-Markovian nature. Consider the following LOMDP $\mathbb{M}$

$$\begin{array}{lcll} 1: & \mathtt{q} & \xleftarrow{1.0:0.0:\mathtt{a}} & \mathtt{p}, \mathtt{q}. \\ 2: & \emptyset & \xleftarrow{1.0:1.0:\mathtt{a}} & \mathtt{p}. \\ 3: & \mathtt{p} & \xleftarrow{1.0:0.0:\mathtt{a}} & \emptyset. \end{array}$$

and the abstraction level $\mathbb{L} = \{\mathtt{p}, \mathtt{q}, \emptyset\}$. The induced MDP $\mathbf{L}$ will assign the same probabilities and rewards to the transitions from l_2 to l_1 and from l_3 to l_1. Consequently, the values for l_2 and l_3 are the same in $\mathbf{L}$ as the next state is the same namely l_1, but $\mathbb{M}$ assigns different values to both.

The example shows that a learner following $\mathbf{L}$ has imperfect and incomplete perception of the states of $\mathbf{M}(\mathbb{M})$. This is interesting because $\mathbf{L}$ corresponds to leafs of a first order decision tree used in relational reinforcement learning [Džeroski *et al.*, 2001]. Unfortunately, complete observability is necessary for learning methods based on MDPs. Thus in general, we must use techniques for solving *partially observable* MDPs, see e.g. [Kaelbling *et al.*, 1996]. In the present paper, we follow the most naive approach to deal with partially observability, namely ignoring it. That is, we treat the induced MDP $\mathbf{L}$ as if it would be the correct underlying MDP.

5 LQ-Learning

In principle, any known algorithm for computing an optimal policy for $\mathbf{L}$ can be used. There are only two complications. First, the probability function μ is not given. This problem can however be solved using stochastic iterative dynamic programming, i.e. model-free approaches. Second, we do not want to construct $\mathbf{L}$. Instead, we directly want to use $\mathbb{L}$. Below, we sketch LQ learning, which learns the Q function of $\mathbf{L}$ using this idea combined with traditional Q learning. Similar, other methods such as MC, SARSA and actor-critic methods can be adapted.

Logical Q Learning
Let $\mathbb{L}$ be an abstraction level
Initialize $\widehat{Q}_0(\mathbb{L}, \mathtt{a})$ arbitrarily for each $\mathbb{L} \in \mathbb{L}$
n=1
Repeat (for each episode)
  Initialize ground state $Z \in S_{\mathbf{M}(\mathbb{M})}$
  Repeat (for each step in episode)
    Choose action a in Z based on $\widehat{Q}_{n-1}$, cf. (4)
    Let $\mathtt{a}$ be the abstract action corresponding to a
    Take action a, observe r and successor state Z'
    Let $\mathbb{L} \in \mathbb{L}$ (resp. $\mathbb{L}' \in \mathbb{L}$) be the unique abstract state matching Z (resp. Z')
    $\alpha_n := (1 + \text{visits}_n(\mathbb{L}, \mathtt{a}))^{-1}$
    $\widehat{Q}(\mathbb{L}, \mathtt{a})_n := (1 - \alpha_n) \cdot \widehat{Q}_{n-1}(\mathbb{L}, \mathtt{a})$
      $+\alpha_n \cdot (r + \lambda \cdot \max_{\mathtt{a}'} \widehat{Q}_{n-1}(\mathbb{L}', \mathtt{a}'))$
    Set $Z := Z'$ and $n := n + 1$
  Until Z is terminal

Here, $\text{visits}_n(\mathbb{L}, \mathtt{a})$ is the total number of times the abstract state – abstract action pair has been visited up to and including the n-th iteration. $\widehat{Q}(\mathbb{L}, \mathtt{a})_n$ is the approximation of $Q(\mathbb{L}, \mathtt{a})$ after n iterations. To select an action a, we first probabilistically select an abstract action $\mathtt{a}$ in a state $\mathbb{L}$ so that the probability $P(\mathtt{a}|\mathbb{L})$ of selection $\mathtt{a}$ is proportional to $\widehat{Q}(\mathbb{L}, \mathtt{a})_n$, e.g.

$$P(\mathtt{a}|\mathbb{L}) = \frac{T^{\widehat{Q}_n(\mathbb{L}, \mathtt{a})}}{\sum_j T^{\widehat{Q}_n(\mathbb{L}, \mathtt{a}_j)}} \tag{4}$$

with $T > 0$. This is common in Q learning. Then, we select uniformly among all possible ground action given by $\mathtt{a}$ and Z to get a.

Let us now argue that LQ learning converges with respect to $\mathbf{L}$. Each selection of a ground state Z selects a unique state l_i in $\mathbf{L}$. Likewise, when we have observed Z', this uniquely specifies a state l_j. The rewards are stochastic, but they depend on Z and a only. Therefore, the convergence theorem for Q-learning for finite (nondeterministic) MDPs applies to $\mathbf{L}$, cf. [Watkins and Dayan, 1992; Jaakkola *et al.*, 1994]. Moreover, it might be the case that LQ learning can do even better. The equality $V_{\boldsymbol{\pi}}(\mathbb{L}_i) = V_\pi(l_i)$ seems to hold if for each legal trace of $\mathbf{L}$ we can find a legal trace within $\mathbf{M}(\mathbb{M})$. Due to the abstraction, LQ learning should generalize well even in unseen ground states.

6 Experiments

We implemented LQ learning using the Prolog system Sicstus-3.9.0. Our task was to learn an abstract policy for the *stack* LOMDP (see above). This task was motivated by the experiments in relational reinforcement learning (RRL) [Džeroski *et al.*, 2001] and by the fact that the blocks world is the prototypical toy domain requiring relational representations. One of the key differences with the experiments reported by [Džeroski *et al.*, 2001] is that we exclusively use the standard predicates `on`, `cl`, and `bl`. [Džeroski *et al.*, 2001] also needed to make use of several background knowledge predicates such as `above`, `height` of stacks as well as several directives to the inductive logic programming function approximator in order to be able to learn adequate policies. Another difference to our approach is that RRL induces the relevant abstract states automatically using a regression tree learner.

The discount factor was 0.9, and the temperature T to select an action was increased by 1.004 each epoch starting with 1.0. Therefore, the agent favors exploration during

[5] A nondeterministic MDP can be converted into a deterministic one. Maximizing the expected future reward depends only on the expected reward in each state, and not on the probability distribution over rewards. In our case, $R(l_i, a, l_j) := \sum_{\mathbb{H} \xleftarrow{p:r:\mathtt{a}} \mathbb{B} \in \mathbb{T}} \mu([\mathbb{B}]|[\mathbb{L}_i]) \cdot p \cdot \mu([\mathbb{L}_j]|\mathbf{S}(\mathbb{H})) \cdot r$ would do.

early states of learning, then gradually shifts towards a strategy of exploration. We randomly generated 10 blocks world states for 4 blocks, 20 for 6 blocks, 30 for 8 blocks, and 50 for 10 blocks using the procedure described by [Slaney and Thiébaux, 2001]. Note that for 10 blocks a propositional MDP would have to represent 58.941.091 states of which 3.628.800 states are goal states. Then, we ran LQ learning on these starting states in order $4, 6, 8$ and 10 blocks. The initial Q function was

$$Q\left(\left\{\begin{array}{c}\texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ \texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)},\texttt{bl(B)},\texttt{bl(D)}\end{array}\right\},\texttt{mv\_fl(A)}\right)=0.0$$
$$Q\left(\left\{\begin{array}{c}\texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ \texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)},\texttt{bl(B)},\texttt{bl(D)}\end{array}\right\},\texttt{mv(A,C)}\right)=0.0$$
$$Q\left(\left\{\begin{array}{c}\texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ \texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)},\texttt{bl(B)},\texttt{bl(D)}\end{array}\right\},\texttt{mv(A,E)}\right)=0.0$$
$$Q\left(\left\{\begin{array}{c}\texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ \texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)},\texttt{bl(B)},\texttt{bl(D)}\end{array}\right\},\texttt{mv(E,A)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(C,D)},\texttt{cl(A)},\texttt{cl(C)}\},\texttt{mv\_fl(A)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(C,D)},\texttt{cl(A)},\texttt{cl(C)}\},\texttt{mv(A,C)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(E,fl)},\texttt{cl(A)},\texttt{cl(E)}\},\texttt{mv\_fl(A)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(E,fl)},\texttt{cl(A)},\texttt{cl(E)}\},\texttt{mv(A,E)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(E,fl)},\texttt{cl(A)},\texttt{cl(E)}\},\texttt{mv(E,A)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{cl(A)}\},\texttt{stop}\right)=0.0$$
$$Q\left(\{\texttt{cl(A)},\texttt{cl(B)}\},\texttt{mv(A,B)}\right)=0.0$$

where we omitted the `absorb` state in front. The whole experiment was repeated 5 times (including sampling the starting states). In all 5 runs, the learned policy (which is optimal at the given abstraction level) was:

$$\begin{array}{rcl}\texttt{mv\_fl(A)} & \leftarrow & \texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ & & \quad\texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)}.\\ \texttt{mv\_fl(A)} & \leftarrow & \texttt{on(A,B)},\texttt{on(C,D)},\texttt{cl(A)},\texttt{cl(C)}.\\ \texttt{mv(E,A)} & \leftarrow & \texttt{on(A,B)},\texttt{on(E,fl)},\texttt{cl(A)},\texttt{cl(E)}.\\ \texttt{mv(A,B)} & \leftarrow & \texttt{cl(A)},\texttt{cl(B)}.\end{array}$$

The learned policy is interesting for many reasons. First, it uniquely specifies a deterministic policy for ground states. Second, it is well known in the planning community [Slaney and Thiébaux, 2001]. It is called *unstack-stack* strategy because it amounts to putting all misplaced blocks on the table and then building the goal state by stacking all blocks from the floor onto one single stack. The total number of moves is at worst twice the optimal. Third,*unstack-stack* perfectly generalizes to all other blocks worlds, no matter how many blocks there are. Finally, it cannot be learned in a propositional setting because here the optimal policy would encode the optimal number of moves.

RRL has learned another policy ("move a block to the highest stack") than LQ learning. However, as argued above, this policy can only be described using additional background predicates, which are not needed in our approach. We believe that RRL would have difficulties in learning the unstack-stack policy using only the predicates `on`, `cl` and `bl`.

Rerunning the experiments with a simpler abstract Q function, omitting the first four abstract values, yields the *unstack-stack* policy, too, but the learning epochs were faster proceeded due to the higher abstraction.

7 Related Work

Within reinforcement learning (RL), there is currently a significant interest in using rich representation languages. [Finney *et al.*, 2002] investigated propositionalization methods in relational domains. They experimentally studied the intermediate language of *deictic representations* (DRs). DRs avoid enumerating the domain by using variables such as *the-block-on-the-floor*. Although DRs have led to impressive results [McCallum, 1995; Whitehead and Ballard, 1991], [Finney *et al.*, 2002]'s results show that DR may also degrade learning performance within relational domains. According to [Finney *et al.*, 2002], *Relational reinforcement learning* (RRL) [Džeroski *et al.*, 2001] is one way to effective learning in domains with objects. RRL is a combination of RL and inductive logic programming (ILP) [Muggleton and De Raedt, 1994]. The key idea is that the Q function is approximated using a relational regression tree learner. Although the experimental results are interesting, RRL has failed to explain – in theoretical terms – why RRL works. Some new insights on this have been obtained.

From a more general point of view, our approach is closely related to *decision theoretic regression* (DTR) [Boutilier *et al.*, 2000]. Here, state spaces are characterized by a number of random variables and the domain is specified using logical representations of actions that capture the regularities in the effects of actions. Because 'existing DTR algorithms are all designed to work with *propositional* representations of MDPs', [Boutilier *et al.*, 2001] proposed *first order DTR* which is a probabilistic extension of Reiter's *situation calculus*. The language is certainly more expressive than that of LOMDPs. However, it is also much more complex. Furthermore, [Boutilier *et al.*, 2001] assume that the model is given whereas in the present paper traditional model-free learning methods have been apply.

The idea of solving large MDP by a reduction to an equivalent, smaller MDP is also discussed e.g. in [Dearden and Boutilier, 1997; Givan *et al.*, 2003; Ravindran and Barto, 2002]. However there, only finite MDPs and no relational or first order representations have been investigated. Furthermore, there has been great interest in abstraction on other levels than state spaces. Abstraction over time [Sutton *et al.*, 1999] or primitive actions [Dietterich, 2000; Andre and Russell, 2001] are useful ways to abstract from specific subactions and time. This research is orthogonal and could be applied to LOMDPs in the future.

Finally, [Baum, 1999] reports on solving blocks worlds with up to 10 blocks using RL related techniques. However, the introduced language is domain-dependent and does not incorporate logic programming.

8 Conclusions

We have presented a representation framework that integrates Markov decision processes with logic programs. This frame-

work allows one to compactly and declaratively represent complex (relational) Markov decision processes. Using functors they might even be infinite. Furthermore, we have introduced abstract policies for LOMDPs and studied their properties. We have shown that their value functions cannot generally be learned using MDP techniques. However, the experiments with a simple upgrade of Q-learning have shown that even naive strategies to handle partially observability can sometimes work. The authors hope that this framework will be useful as a starting point for further theoretical developments in relational reinforcement learning.

Acknowledgements

The authors are deeply grateful Bob Givan for pointing out problems with an earlier version of this paper and for providing the example LOMDP at the end of Section 4.

References

[Anderson *et al.*, 2002] C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov Models and their Application to Adaptive Web Navigation. In D. Hand, D. Keim, O. R. Zaïne, and R. Goebel, editors, *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, pages 143–152, Edmonton, Canada, 2002. ACM Press.

[Andre and Russell, 2001] D. Andre and S. Russell. Programmable reinforcement learning agents. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 1019–1025. MIT Press, 2001.

[Baum, 1999] E. B. Baum. Towards a Model of Intelligence as an Economy of Agents. *Machine Learning*, 35(2):155–185, 1999.

[Boutilier *et al.*, 2000] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121, 2000.

[Boutilier *et al.*, 2001] C. Boutilier, R. Reiter, and B. Price. Symbolic Dynamic Programming for First-order MDPs. In B. Nebel, editor, *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 690–700, Seattle, Washington, USA, 2001. Morgan Kaufmann.

[Dearden and Boutilier, 1997] R. Dearden and C. Boutilier. Abstraction and approximate decision theoretic planning. *Artificial Intelligence*, 89(1):219–283, 1997.

[Dietterich, 2000] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[Džeroski *et al.*, 2001] S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine Learning*, 43(1/2):7–52, 2001.

[Finney *et al.*, 2002] S. Finney, N. H. Gardiol, L. P. Kaelbling, and T. Oates. The thing that we tried didn't work very well: Deictic representation in reinforcement learning. In *Proceedings of the Eighteenth International Conference on Uncertainty in Artificial Intelligence (UAI-02)*, 2002.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In T. Dean, editor, *Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence (IJCAI-99)*, pages 1300–1309, Stockholm, Sweden, 1999. Morgan Kaufmann.

[Givan *et al.*, 2003] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 2003. (in press).

[Hanks and McDermott, 1994] S. Hanks and D. V. McDermott. Modelling a dynamic and uncertain world I: Symbolic and probabilisitic reasoning about change. *Artificial Intelligence*, 66(1):1–55, 1994.

[Jaakkola *et al.*, 1994] T. Jaakkola, M. I. Jordan, and S. P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6), 1994.

[Kaelbling *et al.*, 1996] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research (JAIR)*, pages 237–285, 1996.

[Kersting and De Raedt, 2001] K. Kersting and L. De Raedt. Towards Combining Inductive Logic Programming with Bayesian Networks. In Céline Rouveirol and Michèle Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming*, volume 2157 of *LNAI*, pages 118–131. Springer, 2001.

[Kersting *et al.*, 2003] K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards discovering structural signatures of protein folds based on logical hidden markov models. In R. B. Altman, A. K. Dunker, L. Hunter, T. A. Jung, and T. E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing*, pages 192 – 203, Kauai, Hawaii, USA, 2003. World Scientific.

[McCallum, 1995] A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden States*. PhD thesis, Department of Computer Science, University of Rochester, 1995.

[Muggleton and De Raedt, 1994] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19(20):629–679, 1994.

[Muggleton, 1996] S. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.

[Puterman, 1994] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[Ravindran and Barto, 2002] B. Ravindran and A. G. Barto. Model Minimization in Hierarchical Reinforcement

Learning. In *Proceedings of the Fifth International Symposium on Abstraction, Reformulation and Approximation (SARA-02)*, volume 2371 of *LNCS*, pages 196–211, Kananaskis, Alberta, Canada, 2002. Springer.

[Slaney and Thiébaux, 2001] J. Slaney and S. Thiébaux. Blocks World revisited. *Artificial Intelligence*, 125:119–153, 2001.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[Sutton *et al.*, 1999] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[Watkins and Dayan, 1992] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279 – 292, 1992.

[Whitehead and Ballard, 1991] S. D. Whitehead and D. H. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45 – 83, 1991.

First-Order Probabilistic Models for Information Extraction

Bhaskara Marthi
Computer Science Div.
University of California
Berkeley, CA 94720-1776
bhaskara@cs.berkeley.edu

Brian Milch
Computer Science Div.
University of California
Berkeley, CA 94720-1776
milch@cs.berkeley.edu

Stuart Russell
Computer Science Div.
University of California
Berkeley, CA 94720-1776
russell@cs.berkeley.edu

Abstract

Information extraction (IE) is the problem of constructing a knowledge base from a corpus of text documents. In this paper, we argue that first-order probabilistic models (FOPMs) are a promising framework for IE, for two main reasons. First, FOPMs allow us to reason explicitly about entites that are mentioned in multiple documents, and compute the probability that two strings refer to the same entity — thus addressing the problem of *coreference* or *record linkage* in a principled way. Second, FOPMs allow us to resolve ambiguities in a text passage using information from the whole corpus, rather than disambiguating based on local cues alone and then trying to merge the results into a coherent knowledge base. This paper presents a comprehensive FOPM for a bibliographic database, and explains how the desired inference patterns emerge from the model.

1 Introduction

1.1 Information extraction

Information extraction (IE) is the problem of constructing a knowledge base from a corpus of text documents. Some IE systems extract information from ordinary English prose: for instance, the Message Understanding Conferences [DARPA, 1998] have evaluated systems that extract information about changes of corporate management, airplane crashes, and rocket launches from *Wall Street Journal* articles. Other systems extract information that is presented in highly formatted headers, lists, and tables rather than in complete sentences. For instance, Citeseer [Lawrence *et al.*, 1999a] and Cora [McCallum *et al.*, 2000b] build databases of academic publications; FlipDog [Cohen *et al.*, 2000a] builds a database of job openings from companies' employment web pages; and Froogle [Google Inc., 2003] builds a database of product offers from online stores.

Natural language prose is notoriously ambiguous, and even highly formatted documents (such as web pages listing job openings) can be hard to interpret automatically. An even harder task is combining information from multiple documents into a single coherent knowledge base. In this paper, we argue that first-order probabilistic models (FOPMs) are a promising framework for IE. Because FOPMs allow us to explicitly represent uncertainty about how many objects are in the world and what relations hold between them, we can use a single probabilistic model for everything from parsing or segmenting the text, to inferring object attributes, to inferring relations between objects.

1.2 Advantages of a comprehensive model

One advantage of using such a comprehensive probabilistic model is that we can reason explicitly about *identity uncertainty* — for instance, whether two citations refer to the same publication. This problem has been treated extensively in natural language processing under the name *coreference resolution*, but methods for resolving coreference across documents remain mostly heuristic. In the bibliography domain, resolving identity uncertainty is important both to avoid having duplicate entries for publications and authors in our final database, and so we can assemble more complete descriptions of publications and authors from multiple citations.

A further advantage of having a comprehensive probabilistic model is that we can use cross-document information to disambiguate text. For example, suppose we see a citation that begins, "Wauchope, K. Eucalyptus: Integrating Natural Language Input with a Graphical User Interface". Is "Eucalyptus" part of the title, or is it the author's middle name? If we see other similar citations where the formatting clearly indicates that "Eucalyptus" is part of the title, then the most likely explanation is that all these citations refer to a single publication with "Eucalyptus" in the title, rather than there being two publications, one with "Eucalyptus" in the title and one without. Conversely, if we see another paper by "K. E. Wauchope", it is more likely that "Eucalyptus" is a middle name. As discussed in Section 3.2, a FOPM for the bibliography domain allows this kind of cross-citation disambiguation. Such disambiguation would not be possible if we just chose the most likely segmentation for each citation based on local cues, and passed these results to another layer of the system for merging into a coherent database. That is, processes that are normally bottom-up and opaque to the higher levels of the systems should instead be *cognitively penetrable*, to borrow a phrase from [Pylyshyn, 1984].

1.3 Knowledge base functionality

Once we have created a knowledge base, what would we like to do with it? One application is allowing a user to browse the data and follow hyperlinks between entities: for instance, from a paper, to one of its authors, to other papers by that author. We would also like to support queries about an entity's attributes, such as an author's full name or the page numbers of a journal paper. Finally, we would like to support structured search queries, like "Find all papers by Mike Jordan in UAI '97". One possible answer to such a query is "the system has not seen any citations to such a paper". However, we would like our system to distinguish between the case where it has simply not seen any evidence for the existence of such a paper, and the case where it is very sure no such paper exists—perhaps because it has parsed Mike Jordan's publications page (or the UAI '97 conference program) and seen no such paper. Thus, our knowledge base will need to do more than just store lists of known entities and their attributes.

1.4 Paper overview

Pasula et al. [Pasula *et al.*, 2003] have already applied a FOPM to the bibliography domain. However, that paper discusses a simple model where the only entities are publications and authors, and results are reported only for resolving coreference among citations. The purpose of this paper is to bring the general IE problem to the attention of the FOPM community, and to show how a FOPM can serve as a comprehensive model for an IE task. We use the bibliography domain as our example, but we believe the advantages of a FOPM for coreference resolution and joint disambiguation will be even more important in more complex domains.

We do not assume any particular representation language for the FOPM in this paper. Instead, we focus on the properties of the model itself, particulary how it supports the kinds of reasoning discussed above. Our notation is based on that used in relational probability models (RPMs) [Pfeffer, 2000], but we are not concerned about whether all the complexities of the model can be expressed by an RPM. Later in the paper, we briefly discuss features that would be desirable in a first-order probabilistic language for specifying IE models.

2 Model for the Bibliography Domain

In this section, we describe our probabilistic model of the citation domain. The model, which is an expanded version of the one presented in [Pasula *et al.*, 2003], includes several classes of objects – authors, publications, collections, citation groups, and citations – and its possible worlds consist of the objects and their attributes and relations.

We do not discuss inference or learning in this section, and indeed, exact inference in the model is probably intractable. However, rather than building many approximating assumptions into the model itself, we choose to make the model as rich as possible and perform any approximations during inference. The parameters will be learnt either using Monte-Carlo EM [Tanner and Wei, 1990] or using supervised methods.

2.1 Classes and attributes

Our model has the following generative structure. First, the set of Author objects, and the set of Collection objects are generated independently. Next, the set of Publication objects is generated conditional on the Authors and Collections. After this, CitationGroup objects are generated conditional on the Authors and Collections, and finally, Citation objects are generated from the CitationGroups. We now describe each of these parts in more detail.

Authors

The number of authors who write papers in this field is chosen from a slowly decreasing log-normal prior. Each Author object has an attribute name, which is chosen from a mixture of a letter bigram distribution with a distribution that chooses from a set of commonly occurring names. There is also a multinomial attribute area, which specifies the field this author usually writes papers in (to be more realistic, we could also have multiple such attributes).

Publications

Each publication has attributes area and type which are chosen according to multinomial distributions. Example types include books, conference papers, and journal papers (alternatively, we could have subclasses of publication corresponding to each type, in which case there would be 'class uncertainty'). Publications also have a compound attribute authorList, generated as follows: first, the length of the list is chosen. Next, for each position i in the list, a reference attribute authorList[i] is chosen (by *reference attribute*, we mean an attribute whose value is another object). Most of the time, this attribute is chosen uniformly from the set of authors whose area attribute equals this publication's area, but there is also some probability of choosing uniformly from all the authors. The attribute title is generated from an n-gram model, conditioned on area (this captures the fact that each area has its own commonly used technical terms).

If the publication is of a type that is usually part of a larger collection, such as a conference paper, the collection reference attribute is set, again depending on area, and date and publisher are set to equal collection.date and collection.publisher, respectively. If not, date is generated from a prior distribution, and publisher is chosen uniformly from the set of publishers. A publication may also have other attributes, such as a number for a technical report, which are chosen using appropriate prior distributions.

Publishers

This class has name and city attributes. Instances for the commonly used publishers are included as evidence, and there is a prior that allows for previously unseen publishers.

Collections

A Collection is a journal issue, a book of conference proceedings, or a book that is a collection of articles. It has string attributes name and date, a multinomial attribute type, and a reference attribute publisher.

Citation Groups

Citations often occur in groups. Examples include a reference list at the end of a paper, a bibliography on a particular topic, the publications section of a researcher's homepage, or the table of contents of conference proceedings. The CitationGroup class captures

some of the structure present in these groups. To begin with, there is is an attribute type, which takes values in {refList, bibliography, tableOfContents, homePage, other}. Next, there is a multinomial attribute style, depending on type, that selects from a dictionary of common bibliography styles (there will also be an 'other' style, to model styles that are not in the dictionary).

The CitationGroup class also contains a compound variable publicationList, which is a list of Publication objects. If type $\in$ {refList, other}, this is generated by picking the list length and then sampling independently from a uniform distribution over the publications. If type = bibliography, then the CitationGroup has an area attribute and we sample only from publications with the same area value.

If type = homePage (the case of tableOfContents is analogous), then there is a reference attribute author and a Boolean attribute exhaustive. If exhaustive, then publicationList is the set of Publication objects p such that p.author = author. If not, we need a model for selecting a subset of this set (we assume that there is no repetition within such lists). A simple way to do this is to independently include each member with some probability θ, but more complicated distributions are possible, for example to list only publications before a certain date.

Finally, this class contains a compound variable citationList, of the same length as publicationList. The elements of this list are Citation objects, and each element depends on the corresponding element in publicationList, in a manner specified in the next section.

Citations

A citation is generated conditional on the cited publication, which is the value of the citation's pub attribute. In any CitationList object ℓ, we require that $\ell.\mathsf{citationList}[i].\mathsf{pub} = \ell.\mathsf{publicationList}[i]$. A Citation object also has several 'as cited' attributes that correspond to how the true attributes of the publication are 'corrupted' while creating this citation. As an example, the conditional distribution of titleAsCited given pub.title includes probabilities of misspelling based on edit distance, of abbreviating common technical terms (e.g. "HMM"), and of dropping words like "the". Once again, we have an elementwise dependency between two lists, this time between authorsAsCited and pub.authorList.

There is also an attribute parse that specifies how the various parts are ordered to produce the citation text. It depends on the style attribute of the containing citation list, as well as on pub.type and, if necessary, pub.collection.type (since, for example, journal articles are usually cited differently from conference papers). We use a PCFG for this, but other models such as HMMs are possible.

Finally, there is an attribute text, which will usually be observed. This attribute has a deterministic distribution, which involves filling in the structure found in parse with the text of the asCited attributes.

2.2 Examples

We have specified a rich probabilistic model of the citation domain, but this richness comes at a computational cost. We now argue that this cost is justified, by giving some examples where the model leads to plausible conclusions that would be difficult to reach using simpler methods. Of course, empirical tests would be needed to make the argument conclusive.

In Figure 1, the journal name could potentially refer to either *Journal of Artificial Intelligence Research*, or *Artificial Intelligence Journal*. Suppose the model has previously come across the table of contents for *AIJ* 1996, which is known to be an exhaustive list. None of the citations in that list resembles this one, and so the model would yield a low probability for the hypothesis that one of those papers produced this citation. If the model has not seen an exhaustive list for *JAIR*, it is free to hypothesize the existence of a paper from *JAIR* 1996 whose title is very similar to this one, and would conclude that the paper was published in *JAIR*.[1]

In Figure 2, the model would assign high probability to the event of the citations referring to the same publication, as they have the same title and year of publication. As a result, information from both citations will be combined when inferring the attributes of the underlying publication — the first citation contains the correct conference name, while the second one contains the author's full name, which could be useful if there are other Hegers in the knowledge base.

3 Properties of the Model

3.1 Handling identity uncertainty

One desirable property of our model is that it allows us to reason explicitly about whether two citations refer to the same publication, or whether two papers are written by the same author. For example, although the two citations in Figure 2 look different, we are quite sure they refer to the same publication. In this section, we explain how our model can yield the same conclusion.

A simple scenario

To build intuition, we begin with a very simple scenario, isomorphic to the "balls in an urn" example in [Russell, 2001]. Suppose a library contains n books $b_1, \ldots, b_n$. For now, the only attribute of a book that we will consider is its title: for any b_i, let $P(b_i.\mathsf{title} = x) = P_X(x)$. We create a citation list by repeatedly selecting a book uniformly at random from the library, writing down its title (with some probability of making an error), and returning the book to the shelf. For any citation c, let $P(c.\mathsf{text} = y \mid c.\mathsf{pub}.\mathsf{title} = x) = P_Y(y|x)$. Thus, P_Y models the process by which titles are corrupted as we write them down.

Now suppose we are looking at a citation list with two citations c_1 and c_2, whose text strings are y_1 and y_2. We have two hypotheses about whether the citations refer to the same book:

$$H_1 : \quad c_1.\mathsf{pub} = c_2.\mathsf{pub}$$
$$H_2 : \quad c_1.\mathsf{pub} \neq c_2.\mathsf{pub}$$

We can evaluate the posterior probability that the citations co-refer by comparing the joint probabilities of the two hy-

[1] A third possibility, that this is a previously unseen journal, would be deemed unlikely thanks to the Occam's razor effect discussed in the next section.

```
Helzerman, R. A., and Harper, M. P. 1996.  MUSE CSP: An extension to the constraint
satisfaction problem.  Journal of Artificial Intelligence
```

Figure 1: Disambiguating a journal name

```
Heger, M. (1994).  Consideration of risk in reinforcement learning.  In Proceedings
of the Eleventh International Conference on Machine Learning, pages 105-111, San
Francisco, CA. Morgan Kaufmann.

[Heger, 1994] Heger, Matthias 1994.  Consideration of risk in reinforcement learn-
ing.  In Proceedings of the Machine Learning Conference.  To appear.
```

Figure 2: Combining information from multiple citations

potheses with the evidence:

$$\begin{aligned} p_1 &= P(H_1, c_1.\mathsf{text} = y_1, c_2.\mathsf{text} = y_2) \\ p_2 &= P(H_2, c_1.\mathsf{text} = y_1, c_2.\mathsf{text} = y_2) \end{aligned}$$

Since we choose books uniformly from the n books in the library, the prior probability of H_1 is $1/n$.

$$\begin{aligned} p_1 &= \frac{1}{n} P(c_1.\mathsf{text} = y_1, c_2.\mathsf{text} = y_2 \mid H_1) \\ p_2 &= \frac{n-1}{n} P(c_1.\mathsf{text} = y_1, c_2.\mathsf{text} = y_2 \mid H_2) \end{aligned}$$

To compute $P(c_1.\mathsf{text} = y_1, c_2.\mathsf{text} = y_2 \mid H_1)$, we must sum over all possible values x for $c_1.\mathsf{pub.title}$. To compute $P(c_1.\mathsf{text} = y_1, c_2.\mathsf{text} = y_2 \mid H_2)$, we must sum over both $c_1.\mathsf{pub.title}$ and $c_2.\mathsf{pub.title}$. The results are as follows:

$$p_1 = \frac{1}{n} \sum_x P_X(x) P_Y(y_1|x) P_Y(y_2|x) \quad (1)$$

$$p_2 = \frac{n-1}{n} \left(\sum_{x_1} P_X(x_1) P_Y(y_1|x_1) \right) \left(\sum_{x_2} P_X(x_2) P_Y(y_2|x_2) \right) \quad (2)$$

Occam's razor

So which is greater, p_1 or p_2? Of course, the answer depends on our probability models for book titles and string corruptions, as well as on n. We can gain some insight by considering the case where no string corruption occurs: $P_Y(y_1|x_1) = 1$ if $y_1 = x_1$ and 0 otherwise. Obviously, under this model, H_1 has probability zero when $y_1 \neq y_2$. So suppose $y_1 = y_2 = y$. Then all the terms in the summations where $x \neq y$ are zero, and we have:

$$\begin{aligned} p_1 &= \frac{1}{n} P_X(y) \\ p_2 &= \frac{n-1}{n} P_X(y)^2 \end{aligned}$$

These equations make sense: if H_1 is true, then there is at least one book with title y, but if H_2 is true, there are at least *two* books with title y, so the title probability is squared.

The fact that the title probability is squared in p_2 penalizes H_2 for constraining the values of more hidden variables than H_1 does. The penalty is especially strong because a reasonable prior over publication titles has high entropy: the probability of a typical title might be 10^{-7}. Then if we are selecting from a library of 100,000 books, the posterior probability of H_1 is about 100 times that of H_2. The posterior probabilities only become equal when the library size is about 10^7. Thus, Occam's razor — a preference for hypotheses that explain the observed data using few hidden objects — arises naturally from our model. This effect has been analyzed in the literature on Bayesian model selection since the work of Jeffreys [Jeffreys, 1939]; see [MacKay, 1992] for a more recent overview of the topic.

On the other hand, Occam's razor does not always dominate the computation. Suppose that instead of choosing books from a library and writing down their titles, we are choosing people from a phone book and writing down their first names. The distribution over first names has much lower entropy than the distribution over book titles: for instance, the 1990 census indicated that between 1% and 2% of people in the U.S. were named Mary. So if we select from a phone book with 100,000 entries and get two people named Mary, then p_1 is about 10^{-7} and p_2 is about 10^{-4}: the probability that the two occurrences of Mary are two different people is about 0.999.

String corruptions

Now let us return to the case where the citation text may be an imperfect copy of the book's title. For instance, suppose y_1 = "Doctor Zhivago" and y_2 = "Doctor Zivago". For concreteness, assume $P_X(y_1) = P_X(y_2) = 10^{-7}$; writing "Zhivago" as "Zivago" or vice versa has probability 10^{-3}; and writing the titles correctly has probability close to 1. Also, to make the computations simple, assume all other strings are either extremely unlikely titles, or extremely unlikely to be transcribed as "Doctor Zhivago" or "Doctor Zivago". Then when we substitute into Equations (1) and (2), most of the terms in the summations are near zero, and we can approximate the probabilities as follows:

$$\begin{aligned} p_1 &\approx \frac{1}{n} \left((P_X(y_1) \cdot 1 \cdot 10^{-3}) + (P_X(y_2) \cdot 10^{-3} \cdot 1) \right) \\ &\approx \frac{1}{n} (2 \cdot 10^{-10}) \\ p_2 &\approx \frac{n-1}{n} (P_X(y_1) \cdot 1)(P_X(y_2) \cdot 1) \\ &\approx \frac{n-1}{n} (10^{-14}) \end{aligned}$$

Thus, H_1 has greater posterior probability than H_2 if there are fewer than about 20,000 books in the library. The Occam's razor effect appears here too: H_2 must "pay the cost" of generating each observed title independently, whereas H_1 only "pays" for one title generation and one copying error.

Of course, if y_1 and y_2 are quite different strings, such as "Doctor Zhivago" and "Doctor Dolittle", then the specific set of copying errors necessary to transform one to the other will be less likely than the generation of the title itself, and H_2 will have greater posterior probability.

Unknown numbers of publications

So far, we have assumed the number of books in the library is a known value n. It does not complicate things much to make the number of books a random variable N, with a prior distribution $P_N(n)$. Then, to evaluate hypotheses about coreference, we must sum over the possible values of N. Equations (1) and (2) become:

$$p_1 = \sum_n P_N(n) \left(\frac{1}{n}\right) \sum_x P_X(x) P_Y(y_1|x) P_Y(y_2|x)$$

$$p_2 = \sum_n P_N(n) \left(\frac{n-1}{n}\right) \left(\sum_{x_1} P_X(x_1) P_Y(y_1|x_1)\right) \left(\sum_{x_2} P_X(x_2) P_Y(y_2|x_2)\right)$$

We can also obtain a posterior distribution over N given the observed citations. This involves summing over all possible mappings from citations to publications, as well as summing over publication titles. Formally, let $\mathbf{x} = x_1, \ldots, x_N$ range over assignments of titles to all the publications. Suppose we have seen K citations. Let $\mathbf{y} = y_1, \ldots, y_K$ be the observed titles of the citations and let $\omega = \omega_1, \ldots, \omega_K$ range over mappings from citations to publications. Then $P(N = n|\mathbf{y})$ is proportional to:

$$P_N(n) \sum_{\mathbf{x}} \left(\prod_{i=1}^{n} P_X(x_i)\right) \sum_{\omega} \left(\frac{1}{n}\right)^K \left(\prod_{i=1}^{K} P_Y(y_i|x_{\omega_i})\right)$$

This is analogous to the equation given for balls in an urn in [Russell, 2001]. Intuitively, if we observe the same titles over and over, we will believe there are few books in the library; if we very seldom see the same title twice, we will believe the library is large.

Identity uncertainty in complex models

This section has discussed identity uncertainty in a simplified scenario: writing down the titles of books from a library. Working with the complete bibliography model described in Section 2 introduces two complications. First, the probability models for publication attributes and citation strings are more complex. If c is a citation, then $c.\mathsf{text}$ depends not only on $c.\mathsf{pub.title}$, but also on $c.\mathsf{pub.author}[1].\mathsf{name}$, $c.\mathsf{pub.date}$, $c.\mathsf{pub.collection.name}$, and so on. So to compute the probability that two particular citations co-refer, we need to sum over the possible values of many complex and simple attributes (in practice, we must approximate these sums). Furthermore, two citations of the same publication may differ from each other not because of errors, but simply because they use different formatting and abbreviations.

The second complication is that we are dealing with identity uncertainty for all classes simultaneously: publications, authors, publishers, etc. We may be uncertain not just about whether $c_1.\mathsf{pub.author}[1] = c_2.\mathsf{pub.author}[3]$, but also about whether $c_2.\mathsf{pub}$ even has a third author, and whether $c_2.\mathsf{pub} = c_1.\mathsf{pub}$. We can make sense of all this uncertainty if we think in terms of distributions over logical interpretations (possible worlds). However, these multiple layers of identity uncertainty pose challenges for both representation languages and inference algorithms.

3.2 Cross-citation disambiguation

Another useful property of our model is that it can resolve ambiguities in a citation by using information from other citations. For example, consider the citations in Figure 3. The first citation is ambiguous: it could be that the author's name is K. Eucalyptus Wauchope, or "Eucalyptus" could be part of the paper's title. Of course, a human reader who knew of Kenneth Wauchope and his Eucalyptus system — perhaps from seeing other citations of this paper — would have no trouble seeing that "Eucalyptus" is part of the title. In this section, we show how our model can also disambiguate the first citation using other citations, such as the second one in Figure 3.

Ambiguity given a single citation

To begin with, suppose we observe only the first citation c_1, whose text is y_1. There are two likely hypotheses:

$$A_1 = \begin{cases} c_1.\mathsf{authorsAsCited}[1] = \text{"Wauchope, K."} \\ c_1.\mathsf{titleAsCited} = \text{"Eucalyptus: Integrating..."} \end{cases}$$

$$A_2 = \begin{cases} c_1.\mathsf{authorsAsCited}[1] = \text{"Wauchope, K. Eucalyptus"} \\ c_1.\mathsf{titleAsCited} = \text{"Integrating..."} \end{cases}$$

We can compare the joint probabilities:

$$q_1 = P(A_1, c_1.\mathsf{text} = y_1) = P(A_1) P(c_1.\mathsf{text} = y_1|A_1)$$
$$q_2 = P(A_2, c_1.\mathsf{text} = y_1) = P(A_2) P(c_1.\mathsf{text} = y_1|A_2)$$

Suppose our our title model and our author name model assign about the same probability to an unusual word like "Eucalyptus". Then $P(A_1) \approx P(A_2)$. And if the author-title separator is about equally likely to be a period or a colon, then $P(c_1.\mathsf{text} = y_1|A_1) \approx P(c_1.\mathsf{text} = y_1|A_2)$. So $q_1 \approx q_2$.

Using a second citation

Thus, looking at c_1 alone, a reasonable model assigns equal posterior probabilities to the two hypotheses. But suppose we also observe c_2 (the second citation in Figure 3), whose text is y_2. An ideal model would specify that an institution is unlikely to issue multiple tech reports with the same number: so unless the first publication was issued by some other "NRL" rather than the Naval Research Laboratory, the two citations must co-refer. However, in the model described in Section 2, tech report numbers are chosen independently for each publication. So we must rely on Occam's razor to give high probability to the hypothesis that $c_1.\mathsf{pub} = c_2.\mathsf{pub}$. As shown in Section 3.1, our model prefers this hypothesis because it requires the tech report number (and most of the title) to be generated only once rather than twice.

```
Wauchope, K. Eucalyptus: Integrating Natural Language Input with a Graphical User
Interface. NRL Report NRL/FR/5510-94-9711 (1994).

Kenneth Wauchope (1994). Eucalyptus: Integrating natural language input with a
graphical user interface. NRL Report NRL/FR/5510-94-9711, Naval Research Laboratory,
Washington, DC, 39pp.
```

Figure 3: A pair of citations where the second helps to disambiguate the first.

So most of the posterior probability mass is on worlds where c_1 and c_2 corefer. In y_2, the date is a clear delimiter between the author list and the title, so with probability close to one:

$$\begin{array}{l} c_2.\mathsf{authorsAsCited}[1] = \text{“Kenneth Wauchope”} \\ c_2.\mathsf{titleAsCited} = \text{“Eucalyptus: Integrating...”} \end{array} \quad (3)$$

This is consistent with A_1: if the publication attributes are $c_1.\mathsf{pub.authorList}[1].\mathsf{name} =$ “Kenneth Wauchope” and $c_1.\mathsf{pub.title} =$ “Eucalyptus: Integrating...”, then the c_1 attributes in A_1 and the c_2 attributes in (3) have high probability. Note that this explanation only requires the word “Eucalyptus” to be generated once, as part of the title. On the other hand, if A_2 is true, then “Eucalyptus” occurs in the author name in c_1 and the title in c_2. This is not impossible: it could be that “Eucalyptus” was inserted accidentally in one of the citations; or perhaps both the true title and the true author name include the word “Eucalyptus”, but it was accidentally deleted from the title in c_1. But these explanations are orders of magnitude less likely than the explanation consistent with A_1, so A_1 has greater posterior probability.

Thus, when local cues are insufficient for parsing a citation, our model gives a probability “bonus” to parses that are consistent with the parses of other co-referring citations. Parsing is done as part of the overall inference process, incorporating such top-down information. Note that this approach does not require lists of known author names, paper titles, or journal titles: we are just taking a potentially large set of unlabeled citations and using them to disambiguate each other.

A more difficult example

We must admit that it took some effort to find a citation where the the distinction between authors and title was truly ambiguous. However, there are other domains where fewer formatting cues are available, and word or character n-gram models are less helpful for distinguishing the values of different attributes. As an extreme example, the radio station WPTC displays the artists and titles of songs on its playlist in two unlabeled columns: [2]

```
The Used              Maybe Memories
From Zero             Smack
V Ice                 Nothing is Real
Burnt by the Sun      Soundtrack to the
                        Worst Movie Ever
Tsunami Bomb          Take the Reigns
Squirt                Mr. Normal
```

The reader is challenged to tell which column is which. Clearly, it would help to find other mentions of these artists and titles where their roles are less ambiguous.

[2] http://www.pct.edu/wptc/playlist2.html

4 Desiderata for a FOPL

In section 2, we gave an informal description of our model. Our current implementation essentially requires the details of the model to be hardcoded in. Such an approach will not scale as we build models for many different IE tasks: it would be desirable to have a declarative language for specifying such models. Based on our experience in modeling this domain, here are some of the features we think such a first-order probabilistic language (FOPL) should have:

- A probability distribution over possible worlds which contain objects, functions, and relations.
- Uncertainty about the number of objects in the world, and the ability to make inferences about the existence or nonexistence of objects having particular properties.
- Uncertainty about the relational structure of the world. It is often, as in the citation domain, not possible to specify this structure beforehand.
- The ability to answer queries about all aspects of the world, including the relational and object structure.
- The ability to represent common types of compound objects such as lists and finite sets, and common probability distributions for dependencies between them, such as models for selecting a subset of a set, and models for elementwise dependencies between lists
- The ability to represent probabilistic dependencies that don't have a natural generative structure, such as the dependence between authors, topics, and papers.
- An efficient inference algorithm with provable guarantees on accuracy and computational complexity, and ways to adjust the tradeoff between these two.
- The ability to incorporate domain knowledge into the inference algorithm. For example, in MCMC this knowledge can be used to design a proposal distribution.
- A learning procedure which allows priors over the parameters.

5 Inference

Because exact inference in our model is intractable, we use MCMC [Gilks *et al.*, 1996; Andrieu *et al.*, 2003] as our inference procedure. Specifically, we use a Metropolis-Hastings proposal distribution, the details of which are described in [Pasula *et al.*, 2003]. This proposal includes moves that create and destroy objects, as well as moves that change the attributes of existing objects. This last type of move includes changes to the parse tree of a citation, thus allowing

top-down information to be used to resolve uncertainty about the parse.

An important point is that, for most queries, if an object is not referred to by any other objects in the current state, then we don't need to waste time resampling its attributes. This allows us to reason efficiently about worlds with a large number of unseen papers. However, if we are answering queries like "How many papers has Mike Jordan published at UAI?", we are forced to sample attributes of all papers, and so these queries are more difficult.

Designing efficient general-purpose MCMC algorithms for first-order models remains a challenging open problem. We are investigating several possibilities for speeding convergence. *Query-dependent sampling* is based on the idea that when answering a query that only depends on the marginal distribution of a small subset of the variables, we should focus our sampling near those variables. [Marthi *et al.*, 2002] described how to do this for a specific graph structure, but the idea is more broadly applicable. *Rao-Blackwellization* is a technique that can be used when some of the variables are amenable to exact inference conditional on their Markov blanket. These variables then don't need to be sampled, as we can marginalize them out. Finally, a common approximation technique is to replace a distribution by a reweighted distribution over its k most likely values. This is useful for sampling variables with large domains, such as parse trees.

Besides sampling, the other major family of approximate inference algorithms is that of variational approximations. In the future, we hope to apply generalized variational inference [Xing and Russell, 2003] and generalized belief propagation [Yedidia *et al.*, 2001] in this domain, and compare their performance to MCMC.

6 Related Work

6.1 Existing work in IE

A great deal of work on extracting information from news articles is described in the MUC proceedings (most recently [DARPA, 1998]); examples of work on highly formatted text include [McCallum *et al.*, 2000b; Lafferty *et al.*, 2001; Cohen *et al.*, 2002]. However, most IE work has not focused on combining information from multiple documents. IE researchers have made considerable progress on resolving coreference *within* documents, e.g., between nouns and pronouns; see [Harabagiu *et al.*, 2001] and references therein. There has been less work on cross-document coreference resolution, but [Bagga and Baldwin, 1999] describes a method for detecting mentions of the same event in different news stories, and [Lawrence *et al.*, 1999b; McCallum *et al.*, 2000a] discuss coreference among citations.

There has been considerable work on *record linkage*, the task of finding and merging duplicate entries in databases [Fellegi and Sunter, 1969; Cohen *et al.*, 2000b; Bilenko and Mooney, 2002]. However, record linkage algorithms typically take database tuples as input, while we are starting with unsegmented text. Of course, one could do IE to obtain database tuples and then find duplicates with a record linkage algorithm. But then one would not be able to disambiguate text by finding other mentions of the same entities, as our proposed system does.

Our work can be seen as a fusion of information extraction, which deals with the relationship between facts and text, and data mining, which deals with statistical regularities in the facts themselves. Nahm and Mooney [Nahm and Mooney, 2000] have implemented such a combined system, called DISCOTEX, for extracting information about job openings from newsgroup postings. Their system learns association rules between fields (analogous to our prior model over object attributes) and uses these rules to improve the recall of an IE system. Another example of using domain knowledge to improve IE is the DATAMOLD system [Borkar *et al.*, 2001], which was applied to parsing postal addresses. DATAMOLD has a database of containment relationships between cities, provinces, and countries, and prefers parses that include city-country pairs where the city is known to be in that country. If we used a FOPM for this task, we would hope to infer the geographic relationships while parsing the addresses.

6.2 Bayesian modeling

Another way to think about our probabilistic model would be to say that all the unobserved attributes are parameters of the model: then the prior distributions over these parameters become parameter priors, and the problem of choosing how many hidden objects there are (or computing a posterior distribution over the number of hidden objects) is one of model selection (or model averaging). This Bayesian model selection problem has been tackled, for example, by [Green, 1995] using an MCMC inference method.

Researchers in other branches of AI have used similar models where the observed data is generated by first generating some hidden objects, then generating a correspondence between observations and hidden objects, and finally generating the values of the observations conditioned on their corresponding hidden objects. Applications of such models include robot localization [Anguelov *et al.*, 2002], recovering the 3D structure of an object from multiple images [Dellaert *et al.*, 2003], and finding stochastically repeated patterns (motifs) in DNA sequences [Xing *et al.*, 2003]. However, not all these models are fully Bayesian: [Dellaert *et al.*, 2003] estimate the positions of visual features (corner points, etc.) on objects using maximum likelihood. They note that this strategy is feasible only because they assume that in each image, the mapping from observed features to actual features is one-to-one. Thus, there is no question about the number of hidden objects (features), and no need for the Occam's razor effect provided by a fully Bayesian approach.

7 Conclusions

We have argued that first-order probabilistic models are a useful, probably necessary, component of any system that extracts complex relational information from unstructured text data. We presented an example of such a model for one particular information extraction task. Many desirable features of plausible reasoning, such as a preference for simple explanations and the combination of top-down and bottom-up information, which are lacking in most nonrelational or non-probabilistic IE systems, occur naturally in our model.

Some of the directions we plan to pursue in the future include defining a representation language that allows such models to be specified declaratively, scaling up the inference procedure to handle large knowledge bases, and tackling domains where the observed text is even less structured.

References

[Andrieu *et al.*, 2003] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[Anguelov *et al.*, 2002] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun. Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proc. 18th UAI*, 2002.

[Bagga and Baldwin, 1999] A. Bagga and B. Baldwin. Cross-document event coreference: Annotations, experiments, and observations. In *Proc. ACL-99 Workshop on Coreference and Its Applications*, pages 1–8, 1999.

[Bilenko and Mooney, 2002] M. Bilenko and R. J. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical Report AI 02-296, AI Lab, Univ. of Texas at Austin, 2002.

[Borkar *et al.*, 2001] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In *Proc. ACM SIGMOD Conf.*, 2001.

[Cohen *et al.*, 2000a] W. Cohen, A. McCallum, and D. Quass. Learning to understand the Web. *IEEE Data Engineering Bulletin*, 23(3):17–24, 2000.

[Cohen *et al.*, 2000b] W. W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. In *Proc. 6th KDD*, pages 255–259, 2000.

[Cohen *et al.*, 2002] W. W. Cohen, M. Hurst, and L. S. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *Proc. 11th WWW*, 2002.

[DARPA, 1998] DARPA, editor. *Proc. 7th Message Understanding Conference (MUC-7)*, Fairfax, VA, 1998. Morgan Kaufman.

[Dellaert *et al.*, 2003] F. Dellaert, S. M. Seitz, C. E. Thorpe, and S. Thrun. EM, MCMC, and chain flipping for structure from motion with unknown correspondence. *Machine Learning*, 50:45–71, 2003.

[Fellegi and Sunter, 1969] I. Fellegi and A. Sunter. A theory for record linkage. *JASA*, 64:1183–1210, 1969.

[Gilks *et al.*, 1996] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, 1996.

[Google Inc., 2003] Google Inc. Froogle. http://froogle.google.com, 2003.

[Green, 1995] P. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1995.

[Harabagiu *et al.*, 2001] S. Harabagiu, R. Bunescu, and S. Maiorano. Text and knowledge mining for coreference resolution. In *Proc. 2nd NAACL*, pages 55–62, 2001.

[Jeffreys, 1939] H. Jeffreys. *Theory of Probability*. Clarendon Press, Oxford, 1939.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th ICML*, pages 282–289, 2001.

[Lawrence *et al.*, 1999a] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[Lawrence *et al.*, 1999b] S. Lawrence, C. L. Giles, and K. D. Bollacker. Autonomous citation matching. In *Proc. 3rd Int'l Conf. on Autonomous Agents*, pages 392–393, 1999.

[MacKay, 1992] D.J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[Marthi *et al.*, 2002] B. Marthi, H. Pasula, S. Russell, and Y. Peres. Decayed MCMC filtering. In *Proc. 18th UAI*, pages 319–326, 2002.

[McCallum *et al.*, 2000a] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. 6th KDD*, pages 169–178, 2000.

[McCallum *et al.*, 2000b] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of Internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[Nahm and Mooney, 2000] U. Y. Nahm and R. J. Mooney. A mutually beneficial integration of data mining and information extraction. In *Proc. 17th AAAI*, pages 627–632, 2000.

[Pasula *et al.*, 2003] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS 15*. MIT Press, Cambridge, MA, 2003.

[Pfeffer, 2000] A. Pfeffer. *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford, 2000.

[Pylyshyn, 1984] Z. W. Pylyshyn. *Computation and Cognition: Toward a Foundation for Cognitive Science*. MIT Press, Cambridge, MA, 1984.

[Russell, 2001] S. Russell. Identity uncertainty. In *Proc. 9th Int'l Fuzzy Systems Assoc. World Congress*, 2001.

[Tanner and Wei, 1990] M. A. Tanner and G. C. G. Wei. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *JASA*, 85:699–704, 1990.

[Xing and Russell, 2003] E. P. Xing and S. Russell. On generalized variational inference, with application to relational probability models. Submitted, 2003.

[Xing *et al.*, 2003] E. P. Xing, M. I. Jordan, R. M. Karp, and S. Russell. A hierarchical Bayesian Markovian model for motifs in biopolymer sequences. In *NIPS 15*. MIT Press, Cambridge, MA, 2003.

[Yedidia *et al.*, 2001] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS 13*. MIT Press, Cambridge, MA, 2001.

A Note on the Unification of Information Extraction and Data Mining using Conditional-Probability, Relational Models

Andrew McCallum
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003 USA
mccallum@cs.umass.edu

David Jensen
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003 USA
jensen@cs.umass.edu

Abstract

Although information extraction and data mining appear together in many applications, their interface in most current systems would better be described as serial juxtaposition than as tight integration. Information extraction populates slots in a database by identifying relevant subsequences of text, but is usually not aware of the emerging patterns and regularities in the database. Data mining methods begin from a populated database, and are often unaware of where the data came from, or its inherent uncertainties. The result is that the accuracy of both suffers, and significant mining of complex text sources is beyond reach.

This position paper proposes the use of unified, relational, undirected graphical models for information extraction and data mining, in which extraction decisions and data-mining decisions are made in the same probabilistic "currency," with a common inference procedure—each component thus being able to make up for the weaknesses of the other and therefore improving the performance of both. For example, data mining run on a partially-filled database can find patterns that provide "top-down" accuracy-improving constraints to information extraction. Information extraction can provide a much richer set of "bottom-up" hypotheses to data mining if the mining is set up to handle additional uncertainty information from extraction.

We outline an approach and describe several models, but provide no experimental results.

1 Introduction

Data mining gives us the ability to see patterns, predict the future, and make informed decisions based on the evidence in large databases. For example, data mining of categorical and numerical consumer shopping data allow a retailer to understand which items are bought by the same customers, predict sales of seasonal items, and more efficiently manage its inventory.[1] Over the past decade, the use of data mining techniques has revolutionized many commercial and government enterprises by enabling more accurate decision making in such areas as industrial control [Wang, 1999], fraud detection [Fawcett and Provost, 1997], inventory management [Agrawal *et al.*, 1993], and customer relationship management [Domingos and Richardson, 2001].

There is already much data in the necessary "database-form," (with fields and records), but there is also a vast amount of important information available only in natural language text, such as Web pages, publications, corporate memos, research findings, government reports and other documents. To be accurately mined, these data must first be be first organized and normalized into database-form.

Information extraction aims to do just this—it is the process of filling the fields and records of a database from unstructured text. Its traditional intended use is as the first step of a pipeline in which unstructured text is converted into a structured database, and then data mining produces predictive models from this database. Historically information extraction has most often been studied for news articles [Appelt *et al.*, 1995], but more recently has been applied to many textual formats, including Web pages [Soderland, 1997; Craven *et al.*, 1998; Blei *et al.*, 2002], government reports [Pinto *et al.*, 2003], scientific articles [Lawrence *et al.*, 1999; McCallum *et al.*, 2000b; Ray and Craven, 2001] and legal documents [Bruninghaus and Ashley, 2001]. Also recently there has been somewhat of a revolution in the use of statistical and machine learning methods for information extraction, *e.g.* [Bikel *et al.*, 1997; McCallum *et al.*, 2000a; Lafferty *et al.*, 2001; Carreras *et al.*, 2002; Roth and tau Yih, 2002; Ray and Craven, 2001; Klein *et al.*, 2003].

However, in spite of the improved results of these machine learning methods, and in spite of a surge of over-anxious commercial ventures claiming success, information extraction with sufficient accuracy to dump directly into data mining remains elusive, and the promise of mining from textual sources is largely unfulfilled. Although there has been much discussion about combining information extraction and data mining, there are few examples of successful pipelining of the two technologies on anything but simple problems.

This position paper proposes *extraction-mining random fields*—a family of models for improving our ability to data mine information in unstructured text by using information extraction and data mining methods that have such tight integration that the boundaries between them disappear, and they

[1] While in some circles, data mining indicates "unsupervised discovery of patterns," here we include classification and other supervised learning tasks within the scope of data mining.

can be accurately described as a *unified framework* for extraction and mining. This framework uses rich, intertwined undirected graphical models in which extraction decisions and data-mining decisions are made with a common inference procedure—the evidence for an outcome being the result of inference both "bottom up" from extraction, and "top down" from data mining. Thus (1) intermediate hypotheses from both extraction and data mining can be easily communicated between extraction and data mining in a closed loop system, (2) mutually-reinforcing evidence and uncertainty will have the opportunity to be properly marshaled, (3) and accuracy and confidence assessment will improve.

Our focus in both areas is on relational data—data about entities and links that is better described by graphs than by the flat attribute-value representations used in much of machine learning. The edges (or hyper-edges) in such graphs represent binary (or n-ary) relations between entities, such as familial relationships among people or hyperlink relations among web pages. In terms of probabilistic models, individual relations or chains of multiple relations help structure the probabilistic dependencies among entities. More formally, in addition to having graph structure, we define a relational task as one in which the system's outputs have several components, $\mathbf{y} = \{y_1, ...\}$, and not all the components are independent from each other given the inputs, $\mathbf{x}$; thus $\exists i, j$ such that $P(y_i|\mathbf{x}) \neq P(y_i|y_j, \mathbf{x})$.

Our proposed models are all trained to maximize conditional probability of the outputs given the inputs. Such models have the advantage of not requiring explicit representation of dependencies among the features of the input. This is especially advantageous when using complex, overlapping and multi-granularity features, as is common in work with natural language text [McCallum *et al.*, 2000a; Lafferty *et al.*, 2001].

2 The Task and Problem

Data mining has enabled a revolution in planning, decision making and organizational efficiency in many areas of industry and government. A similar revolution could be brought about in many additional areas if it were possible to mine the vast amount of information currently locked in unstructured text. In many domains, there is far more information in documents and other text than there is in structured databases.

For example, CiteSeer [Lawrence *et al.*, 1999] mines the Web for research papers, extracts title, authorship and citation information, and thus enables analysis of the citation graph for finding seminal and survey papers. This service has had significant impact on the the practice of computer science research. However, the variety of fields and relations it extracts is small, and the limited accuracy of its existing relations constrains the ability to perform more sophisticated data mining. For example, Pasula *et al.* [2002] note that CiteSeer contains records of over 30 separate AI textbooks written by Russel and Norvig, when actually there is only one.

Unfortunately, the complex data mining of rich unstructured text is not feasible with current methods: extraction is often inaccurate, co-reference resolution is often poor, and data mining is not able to recover from a noisy database.

2.1 Inaccurate extraction

State-of-the-art precision and recall for extracting named entities (such as people, organizations and locations) is in the low- to mid-90's percent for many systems and domains—including BBN's IdentiFinder on news wire articles [Bikel *et al.*, 1997], Cora's hidden Markov models on research paper headers [McCallum *et al.*, 2000b], and WhizBang Lab's extractors on web page data [McCallum, 2002]. The winners of the CoNLL-2002 named entity competition [Carreras *et al.*, 2002] reached only about 80% precision and recall on Spanish newswire text. One of the most recent research papers on named entity extraction from Web pages reached precision and recall in the high 80s [Collins, 2002]. Reaching about 90% precision and recall may seem good until one realizes that this means that more than one in ten fields in the database are either incorrect or missing.

When we consider the accuracy of database records (or "relations") instead of individual fields, the state-of-the-art is even worse. For a relation to be correct, all its constituent fields and its relation-type categorization must be correct. Even if a system had 95% accuracy in extracting individual fields and categorizing relations, the overall accuracy of a three-slot relation would be only 80%. This happens because each automated decision in the formation of a relation is performed independently, and the errors compound. For example, the top performer in the 2002 DARPA ACE evaluation had entity extraction precision and recall scores of about 80%, but binary relation extraction scores of only roughly 60% [DARPA, 2002].

A better solution should not treat the components of a relation independently, but should make coordinated decisions and model them together. For example, the model could know that a person graduates from a university, not from another person, and use this to coordinate its extraction of a person name, a university name, and its categorization of the relation. If done correctly, relations should actually provide constraints that help improve overall extraction accuracy, not hurt it. This idea is one component of our proposed approach, and is expanded in the section 3.

2.2 Poor coreference resolution

One of the key problems in current systems that work on unstructured text is recognizing when two string names are referring to the same entity. For example, "Colin Powell," "Powell," "U.S. Secretary of State," "the Secretary of State" are not string-identical, but in some context may all refer to the same person. If they get separate entries in the database, relational connections will be missing, and data mining will not find the patterns it should [Jensen, 1999].

Coreference (also known as de-duplication, or record matching) is also a difficult problem in traditional databases. There, some of the most successful approaches bring to bear a multitude of evidence from different fields of each record, *e.g.* [Borthwick *et al.*, 2000; Bilenko and Mooney, 2002]. However, the problem is especially difficult in text domains where the original data is unstructured, the availability of some fields is questionable, and the collection of fields into records has not yet been performed.

Often some amount of coreference resolution must happen in order to gather all fields of a record because the infor-

mation is dispersed across multiple sentences, paragraphs or documents. Thus we have a difficult chicken-and-egg problem: to perform accurate coreference we need a multitude of evidence from different fields of a record, but to gather all the fields of a record we rely on coreference resolution. Coreference resolution and record (relation) building should happen simultaneously in a coordinated effort.

Part of the reason coreference has historically been so problematic in text domains is that it sits on the boundary between extraction and data mining. Formation of the fields and records is addressed by extraction; record de-duplication is usually seen as a database issue. However, as we have just pointed out, they rely on each other in highly intertwined ways. They cannot be deeply solved separately. This is particularly true of cross-document coreference, an extremely important problem that has received little attention.

Early work on relational coreference resolution includes Pasula *et al.* [2002] and McCallum and Wellner [2003]; the later is briefly described in section 3.4.

2.3 Fragile data mining

One might hope that data mining techniques could compensate for the errors introduced by inaccurate extraction and poor coreference resolution. Research in data mining has a long history of constructing accurate models using combinations of many features. Work with decision trees, Bayesian classifiers, support-vector machines, and ensemble methods, has produced methods that combine large numbers of (potentially noisy) features into a single model that can "damp out" high levels of noise and allow accurate predictions.

Unfortunately, this existing work on high-accuracy classifiers presumes propositional instances, each of which has large numbers of features. In contrast, data produced by information extraction has a rich relational structure, but each entity and relation has relatively few features. This obviates the strategies used to such great effect in propositional learners, and can often result in brittle, inaccurate models. Some relational learning techniques attempt to overcome this difficulty by constructing relational features to supplement the relatively small number of intrinsic features present in the raw data. However, such calculations rely simultaneously on both extracted relations (the most error prone element of extracted data) and extracted features, so they suffer from the combined errors of both types of data.

Fortunately, relational graphical models can leverage two sources of added power to compensate for the relative lack of high-quality features. First, these models can incorporate information about the uncertainty of the underlying data to influence how strongly specific features influence predictions. By using uncertainty estimates on extracted entities, relations, and features, the models can "play to the strengths" available in extracted data. Second, these models can use the relational structure of the data themselves so that high-confidence inferences about some entities can be used to aid inferences about related entities. We discuss this approach in more detail in section 3.3.

2.4 Consequences of Problems

The consequence of these problems is that little or no data mining is conducted on databases produced through extraction from unstructured text.

A few preliminary research-level exceptions are discussed in section 4. Two larger-scale exceptions are FlipDog.com (a database of job openings populated mostly through extraction), and CiteSeer [Lawrence *et al.*, 1999] (a database of research papers and citations populated through various automatic methods). However, FlipDog makes significant concessions in recall to obtain higher precision, and also relies on non-trivial amounts of human verification to clean its database [McCallum, 2002]. In CiteSeer, the extraction of research paper references is significantly easier than most kinds of named entity extraction from less structured data, and CiteSeer still makes many significant errors in extraction and coreference (as described in the "Russell and Norvig" example in section 2).

We believe that extraction and data mining should be able to help each other through close coordination rather than each failing separately. We describe our approach in some detail in the next section.

3 A Solution

Our approach to both information extraction and data mining is based on statistical machine learning and probabilistic models. These methods have had a high degree of success in each of the two fields recently. There are also strong benefits to using models of IE and data mining that are tightly compatible with each other—with both of them speaking the language of probabilities, they will share a common, low-level communication medium.

In fact, we propose a model that is so tightly integrated that the boundaries between IE and data mining disappear. Our proposed unified system can be understood as a single, large conditionally-trained undirected graphical model. This is a type of probabilistic model that excels at capturing highly interdependent, relational data in which strict causality among events is not necessarily apparent—a set of circumstances appearing both in low-level text data and higher-level relational data mining.

In the next subsections we describe how recent research in both information extraction and data mining have independently arrived at undirected graphical models, and then describe our proposed unification, the advantages of our approach, and several specific models.

3.1 Models for information extraction

Finite state machines are the dominant model for information extraction both in industry and research. There was significant early work with hand-tuned finite state transducers, *e.g.* [Jerry *et al.*, 1996], but more recent work is with finite state machines whose parameters are set by machine learning—most commonly hidden Markov models [Bikel *et al.*, 1997; Leek, 1997; Freitag and McCallum, 1999; Ray and Craven, 2001].

Hidden Markov models have parameters for state-to-state transition probabilities and per-state observation emission probabilities. From these one can easily calculate the probability that the model would have generated a particular state sequence associated with a particular observation symbol sequence. When used for extraction, the emission symbols are

typically natural language words, and states are associated with different extraction fields. For example, to extract person names, the hidden Markov model may have two states, one for person-names, and one for other. To perform extraction on a particular word sequence, one uses the Viterbi algorithm to find the state sequence most likely to have generated the given the observed word sequence, and then designates as person names any words Viterbi that claims were generated while in the person-name state.

A disadvantage of hidden Markov models is that, being generative models of the observation sequence, they are limited in their ability to represent many non-independent, overlapping features of the sequence. In other words, since the observations are *generated* by the model, the model must represent any correlations between features in order to faithfully reproduce them. When there are many correlated features, or complex dependencies among them, (or a desire to capture features at multiple levels of granularity and features of the past and future), this modeling is prohibitively difficult, (and in many cases impossible).

The ability to use arbitrary features is important because often significant features of the observation sequence include not just the identity of the words, (e.g. the word "Wisniewski" was observed), but also other features of the word and its context—for example, it is capitalized, it ends in "ski," it is in bold face, left justified, it is a member of a list of last names from the U.S. Census, the previous word is a recognized first name, and the next word is "said". All of these are powerful pieces of evidence that the word is a person's last name—especially useful evidence if the word "Wisniewski" does not appear anywhere in the labeled training data, (a typical circumstance in the common case of limited labeled data).

Furthermore, and highly significant to our approach, we also want an information extraction model that provides a place for data mining to inject arbitrary "top-down" information that could improve extraction accuracy. A simple, yet powerful interface between data mining and extraction is for the extraction model to see the output of data mining essentially as additional features—top-down features instead of bottom-up word features. Details and variations are discussed in the following subsections.

Maximum entropy Markov models (MEMMs) [McCallum *et al.*, 2000a] and conditional random fields (CRFs) [Lafferty *et al.*, 2001] are two conditional-probability finite state models that—because they are conditional instead of generative—afford the use of arbitrary features in their modeling of the observation sequence.

Conditional Markov models have provided strong empirical success. They extracted question-answer pairs from Frequently-Asked-Question lists with double the precision of an HMM [McCallum *et al.*, 2000a]. They reduced part-of-speech tagging errors on unknown words by 50% over an HMM [Lafferty *et al.*, 2001]. They have achieved world-class results in noun phrase segmentation [Sha and Pereira, 2003a]. They found tables in government reports significantly more accurately than previous methods [Pinto *et al.*, 2003]. They remain an extremely promising area for new research.

3.2 Models for data mining

Work on data mining has traditionally relied on a common family of techniques for learning statistical models from propositional data. For example, algorithms that learn decision trees [Quinlan, 1993; Breiman *et al.*, 1984], linear models [McCullagh and Nelder, 1989], and simple Bayesian classifiers [Mitchell, 1997] are typical parts of many data mining systems. More recently, work has focused on how to combine simple models into more complex models such as ensembles learned through bagging [Breiman, 1996] and boosting [Schapire, 1999]. Finally, the use of graphical models of propositional data [Jordan, 1998] has become widespread, often incorporating simple classifiers such as decision trees to estimate conditional probability distributions.

Unfortunately, attempting to adapt these propositional learners to relational data can lead to serious statistical errors. Over the past two years, the second author has identified several ways in which the structure of relational data can cause significant bias in learned models. For example, many relational data sets exhibit autocorrelation among the features of relational entities (e.g., most coauthors of a paper tend to be employed by a single type of organization). This autocorrelation can be useful for prediction, but it can also systematically bias naive learning algorithms toward features with the *least* supporting evidence [Jensen and Neville, 2002]. More recently, we have also discovered that correlation between the feature values and the structure of relational data can cause naive learners to produce models with invalid structure [Jensen *et al.*, 2003a]. We have found solutions to both these problems [Jensen and Neville, 2003; Jensen *et al.*, 2003a] and incorporated them into our own relational learning algorithms.

Another failing of many traditional data mining techniques is that they do not use uncertainty information on data items. Although we know the probability of correct extraction for a given entity or relation, most data mining models cannot use that information during learning or inference. Notable exceptions are the techniques for learning and inference in graphical models.

A final failing of traditional models learned through data mining is that they make predictions for each instance (e.g., each document) individually, independent of any other. These approaches typically "propositionalize" the data, by flattening complex relational data into a single table. Such approaches miss the potential opportunity to correct for errors on some instances based on higher-confidence predictions about related instances.

Fortunately, a small but growing body of researchers is exploring new methods for relational data mining that overcome these difficulties. These techniques move beyond naive adaptations of methods for propositional learning, and they take seriously the unique opportunities and challenges presented by relational data. One excellent example is the work by Getoor *et al.* [2001] on learning probabilistic relational models (PRMs), a form of directed graphical model that learns the interdependence among features of related entities. PRMs have been applied to learning relationships among movies and their actors, among tuberculosis patients and their contacts, and among Web pages.

Despite their power, PRMs are unable to express many of

the types of mutual dependence among features because a PRM must be a *directed acyclic* graph. For example, the acyclicity constraint makes it nearly impossible to express autocorrelation [Jensen and Neville, 2002], a nearly ubiquitous feature of relational data sets. Autocorrelation can be used to greatly improve model accuracy through the natural feedback of probabilistic inference.

Undirected graphical models, however, remove the acyclicity constraint, and some of the most advanced work in relational learning has focused on these models in the past two years. These models combine the benefits of traditional graphical models, including understandability and incorporation of uncertainty, with the advantages of full inferential feedback. Studies of *relational* or *collective* classification with undirected models [Taskar *et al.*, 2002; Neville and Jensen, 2000] have shown impressive gains in accuracy. Based on the our preliminary work, undirected graphical models of relational data are poised to produce substantial accuracy gains in almost all cases, analogous to the type of gains seen with ensemble classifiers [Breiman, 1996; Schapire, 1999] and for the same reasons—substantial reductions in variance because of an increase in the evidence used for each inference [Jensen *et al.*, 2003b].

3.3 A Unified Model

Thus, conditionally-trained, undirected graphical models are at the heart of recent work in two fields: one examining data at word level for information extraction, and the other examining data at the entity level for data mining. Even though they provide modeling at different levels of abstraction, they meet each other at the entity level, and are fundamentally providing models of the same data—one "bottom up," the other "top down."

The two models are entirely compatible with each other. An undirected graphical model of information extraction can be combined with an undirected graphical model of data mining in one grand, unified graphical model—a unified probabilistic model, with a unified representation of data and outcomes, a unified set of parameters, unified inference procedures, and unified learning procedures.

Seen in this light, information extraction and data mining are not separate processes, but a single collective whole. No hard, brittle decisions need to be made at one stage of a pipeline in order to be passed to the next stage—the subtlest and most uncertain of hypotheses can be communicated back and forth between extraction and data mining, each helping the other converge to an agreed upon conclusion.

For example, consider the following scenario. Word-level features alone might leave ambiguous whether an appearance of the word "Tracy" on a university Web page is a person name or a project name. An appearance of "Beth Smith" on the same page might more certainly be hypothesized to be a person name. Through initial coreference analysis, we might find Beth Smith's home page, and her relations to some other people. These patterns of relations (in combination with the words on her home page) might cause the model to decide that Beth Smith is likely a professor. Knowing this might help provide just enough additional evidence to the extraction model running in the context of the original page that it is able to hypothesize a Principal-Investigator-Of relation between "Beth Smith" and "Tracy". Since the data mining model parameters indicate that this relation only occurs between a person and a project, it can be correctly deduced that the word "Tracy" must be a project name here, not a person name. And furthermore an appearance of the person name "Tracy Jones" on a different Web page can correctly be said not to be co-referent with the project "Tracy" on the first page. All of these constraints are communicated in subtle shades of probability that work themselves out through the statistically principled methods of inference.

3.4 Conditional Random Fields

In this section we define conditional random fields and describe how they may be used to create unified models for information extraction and data mining—illustrating this framework with several specific examples.

Conditional Random Fields [Lafferty *et al.*, 2001] are undirected graphical models (also known as *random fields*) used to calculate the conditional probability of values on designated output variables given values assigned to other designated input variables.[2]

Let $\mathbf{X}$ be a set of input random variables, and $\mathbf{Y}$ be a set of output random variables. Then, by the fundamental theorem of random fields [Hammersley and Clifford, 1971], a conditional random field defines the conditional probability of values $\mathbf{y}$ given values $\mathbf{x}$ to be a product of potential functions on cliques of the graph,

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \prod_{c \in \mathcal{C}} \Phi_c(\mathbf{x}_c, \mathbf{y}_c),$$

where $Z_{\mathbf{x}} = \sum_{\mathbf{y}'} \prod_{c \in \mathcal{C}} \Phi_c(\mathbf{x}_c, \mathbf{y}_c)$ is the partition function (normalizer), $\mathcal{C}$ is the set of all cliques, $\Phi_c(\cdot)$ is the potential function for clique c, $\mathbf{x}_c$ is that sub-set of the variables in $\mathbf{x}$ that participate in clique c, and $\mathbf{y}_c$ is defined analogously. We calculate the potential functions as a log-linear combination of weighted features, $\Phi_c(\mathbf{x}_c, \mathbf{y}_c) = \exp(\sum_k \lambda_{kc} f_{kc}(\mathbf{x}_c, \mathbf{y}_c))$, where $f_{kc}(s_{t-1}, s_t, \emptyset, t)$ is an arbitrary feature function over its arguments, and λ_{kc} is a learned weight for each feature function.

Linear Chain

In the special case in which the designated output nodes of the graphical model are linked only by edges in a *linear chain*, CRFs make a first-order Markov independence assumption among output nodes, and thus correspond to finite state machines (FSMs), which have been shown to be suitable sequence models for information extraction, *e.g.* [Bikel *et al.*, 1997; McCallum and Li, 2003].

Let $\mathbf{x} = \langle x_1, x_2, ...x_T \rangle$ be some observed input data sequence, such as a sequence of words text in a document, (the values on n input nodes of the graphical model). Let $\mathcal{S}$ be a set of FSM states, each of which is associated with a label, $l \in \mathcal{L}$, (such as a label PERSON). Let $\mathbf{y} = \langle y_1, y_2, ...y_T \rangle$ be some sequence of states, (the values on T output nodes).

[2] The term "random field" has common usage in the statistical physics and computer vision communities. In statistics the same models are also known as "Markov networks." Thus *Conditional Markov Networks* [Taskar *et al.*, 2002] are identical to Conditional Random Fields.

CRFs define the conditional probability of a state sequence given an input sequence as

$$P_\Lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\mathbf{x}} \exp\left(\sum_{t=1}^{T}\sum_{k}\lambda_k f_k(y_{t-1}, y_t, \mathbf{x}, t)\right).$$

This model ties parameters $\Lambda = \{\lambda, ...\}$ across sequence positions, but this is just one possible type of tying. Various patterns of parameter tying may be based on arbitrary SQL-like queries [Taskar *et al.*, 2002]. Several specific patterns relevant to unification of extraction and data mining are described below. Many others in this framework are also possible.

Cross-referenced Linear Chain
The previous model captures dependencies between adjacent pairs of labels, but in some cases we may have reason to believe that other, arbitrarily-separated words have dependent labels. For example, capturing the fact that two identical capitalized words in the same document often should share the same label will help us know that "Green" is a last name when we have seen the phrase "David Green" elsewhere in the document. Such dependencies among selected pairs, $\mathcal{P}$, of arbitrarily-separated words can be represented with a *cross-referenced linear chain*,

$$P_\Lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\mathbf{x}} \exp\left(\sum_{t=1}^{T}\sum_{k}\lambda_k f_k(y_{t-1}, y_t, \mathbf{x}, t) + \sum_{\langle t,t'\rangle\in\mathcal{P}}\sum_{k'}\lambda_{k'} f_{k'}(y_t, y_{t'}, \mathbf{x}, t)\right).$$

Note that the edges among the output variables now form loops, and inference is more difficult than before. Approximate inference methods are discussed below.

Factorial Linear Chain
When there are multiple dimensions of labels to be predicted—for example part-of-speech, phrase boundaries, named entities, and the classification of entities into categories (such as STUDENT and PROFESSOR)—these multi-dimensional labels can be simultaneously predicted and efficiently represented in a factorial model. Ghahramani and Jordan [1995] describe a factorial HMM. Factorial CRFs are detailed in [Rohanimanesh and McCallum, 2003], and define the probability of two label sequence factors, $\mathbf{y}$ and $\mathbf{y}'$, connected in a grid as

$$P_\Lambda(\mathbf{y},\mathbf{y}'|\mathbf{x}) = \frac{1}{Z_\mathbf{x}} \exp\left(\sum_{t=1}^{T}\sum_{k}\lambda_k f_k(y_{t-1}, y_t, \mathbf{x}, t) + \sum_{t=1}^{T}\sum_{k'}\lambda_{k'} f_{k'}(y'_{t-1}, y'_t, \mathbf{x}, t) + \sum_{t=1}^{T}\sum_{k''}\lambda_{k''} f_{k''}(y_t, y'_t, \mathbf{x}, t)\right).$$

Affinity- or Relationship-Matrix
When predicting entity coreference (or other types of relationships), rather than a sequence of labels, let the output be a matrix $\mathbf{w} = \{w_{11}, w_{12}, ...w_{tt'}, ...w_{TT}\}$ of labels on pairs of words (or entities), and forming a matrix of coreference decisions or other binary relationships. We define the distribution,

$$P_\Lambda(\mathbf{w}|\mathbf{x}) = \frac{1}{Z_\mathbf{x}} \exp\left(\sum_{t,t'}\sum_{k'}\lambda_{k'} f_{k'}(w_{tt'}, \mathbf{x}, t, t') + \sum_{t,t',t''}\lambda_* f_*(w_{tt'}, w_{t't''}, w_{tt''})\right).$$

This model, in which inference corresponds to graph partitioning, is further described in McCallum and Wellner [2003], where the need for dependencies among the w's in the second sum is also explained. Another variant described there also predicts attributes associated with entities. The matrix can be made sparse by approximation with *Canopies* [McCallum *et al.*, 2000c].

Factorial Chain and Relationship-Matrix
Entity extraction, classification of entities, coreference, and determination of other relationships among entities can all be performed simultaneously by a factorial model over chains and matrices. This is a model that could solve the "Tracy" problem described above. The equation (which we omit to save space) includes a straightforward combination of the sums from the previous two models, plus additional desired dependencies among output variables. Other promising variations include the integration of hierarchical models corresponding to parse trees.

Inference and Parameter Estimation
Given an inference procedure, parameter estimation in all these models can be performed with standard optimization procedures such as conjugate gradient or approximate quasi-Newton methods [Malouf, 2002; Sha and Pereira, 2003b]. Inference for Linear Chain models can be performed efficiently with dynamic programming [Lafferty *et al.*, 2001]. The other models have loops among output variables, and thus we must resort to approximate inference. Approximate inference in the Affinity-Matrix models can be performed by randomized graph partitioning algorithms, as described in [McCallum and Wellner, 2003]. We have had considerable success performing inference in the Factorial Linear Chain [Rohanimanesh and McCallum, 2003] with Tree-based Reparameterization [Jaakkola *et al.*, 2001]. Improved methods of efficient approximate inference in these models remains an open area for research. Feature induction (which also corresponds to graphical structure induction for these models) is described in McCallum [2003].

4 Related Work

There has been a large amount of previous separate work on information extraction and data mining, some of which has been referenced and described previously in this paper.

4.1 Relational extraction and data mining

There is also a new and growing body of work in extraction of *relational* data, as well as separate work in data mining of *relational* data. In extraction, the association of entities into relations has traditionally been performed by classification of entity pairs independently from each other. For example, noun coreference can be decided by the output of a binary maximum entropy classifier indicating whether or not the two nouns in the pair are co-referent [Morton, 1997]. The binary classifiers can also be quite sophisticated, for example using SVMs with complex kernels [Zelenko *et al.*, 2003].

However, these methods perform entity extraction completely independently from association (causing errors to compound), and also make coreference and relation-formation decisions independently from each other (allowing decisions to be inconsistent and errorful). For example, one classifier might decide that "Mr. Smith" is co-referent with "Smith," and another classifier might incompatibly decide that this "Smith" is co-referent with "she." An alternative approach is to extract and build relations in a single augmented finite state machine [Ray and Craven, 2001] or parsing model [Miller *et al.*, 2000], however this only operates over relations formed within one sentence. Other work [Roth and tau Yih, 2002] recognizes and models the dependencies across multiple entity classifications and relations; however it relies on entity extraction having already been performed. Recent work in coreference analysis also explicitly models the dependencies among coreference decisions on multiple pairs of pre-extracted entities [Pasula *et al.*, 2002; McCallum and Wellner, 2003].

As described in section 3.2, there has been a recent surge of research on relational data mining. Particularly notable is work based on undirected graphical models [Taskar *et al.*, 2002], (and also indirectly [Neville and Jensen, 2000]). The former involves experiments on data mining of academic entities, although it does so through Web page and hyperlink classification, not through full information extraction (which would involve extracting multiple sub-segments of text on a page, and more difficult coreference and relation-building analysis).

4.2 Early work in integration of extraction and data mining

There has still been relatively little work on integration between extraction and data mining. Most current work is better characterized as serial juxtaposition, (e.g. [Ghani *et al.*, 2000]), or mining raw text data (such as documents, web sites, hyperlinks, or web logs), (e.g. [Hearst, 1999; Craven *et al.*, 1998; Taskar *et al.*, 2002; Kosala and Blockeel, 2000; Anderson *et al.*, 2002]), but not mining a rich database resulting from information extraction, (that is, sub-segments of text on a page, each referring to different entities—which is significantly more difficult).

One interestingly different approach does not aim to extract a correct database, but instead attempts to data mine a "soft database" consisting of the raw text of each mention, (without any coreference analysis having been performed, and perhaps with extraction boundary errors) [Cohen and Hirsh, 1998]. New database operations, such as "soft joins" may merge records based on TF-IDF similarity instead of exact matches—doing so on the fly in response to a particular query. This approach is intriguing, but it seems only to delay the inevitable difficulties. Much noise and error remains in these soft joins, and this approach could not support complex relational data mining.

Some of the most truly integrated work in extraction and data mining has been done by Ray Mooney's group at UT Austin. For example, in one project, twelve fields of data are extracted from USENET computer-related job ads using a rule learner. The fields include programming-language, hardware-platform, application-area, etc. A second rule learner is applied to an imperfectly-extracted database to produce rules that will predict the value in each field given the others. Then these rules are used to fill in missing values and correct errors in extraction—a very nice example of "closing (one turn of) the loop." This work is a promising first beginning; there remain much additional work to do, especially in the use of stronger statistical machine learning methods, such as graphical models, that have provided world-class performance in other independent extraction and data mining problems. This is the approach we put forward in this paper.

5 Conclusions

We have presented motivation, problems and proposed solutions for a unified framework of extraction and data mining using conditionally-trained undirected graphical models. This approach addresses the three critical topics of integrating extraction and data mining:

Uncertainty management — The hypotheses of both extraction and data mining are represented in probability distributions on nodes of the graphical model. For example, in extraction sections of the model, a node might represent an individual word, and contain a probability distribution over the entity labels person, project, university, other, etc. In the data mining sections of the model, a node might represent a relation between two entities, and contain a probability distribution over the labels principal-investigator-of, adviser-of, project-colleague-of, etc.

With both extraction and data mining embedded in the same model, intermediate hypothesis are naturally communicated back and forth in the language of probabilities. Rather than being a problem, uncertainty becomes an opportunity—with the ability for the intermediate hypotheses of data mining to improve extraction, and vice-versa.

Inferential Feedback — Closed-loop feedback between extraction and data mining is a natural outcome of inference in the unified graphical model.

Note that there has been some previous work on feeding extracted data into data mining (see section 4), and performing inference on this noisy data. However, we are proposing models that actually "close the loop" by feeding results of data mining back into extraction, and looping back to data mining repeatedly. This closed-loop, bi-directional communication will allow subtle constraints to flow both directions, let sharper conclusions be formed by the agglomeration of multiple pieces of uncertain evidence, and help turn the communication of uncertainty into an advantage, not a disadvantage.

Relational Data — Relational data are straightforwardly modeled in undirected graphical models by using tied parameters in patterns that reflect the nature of the relation. Patterns of tied parameters are common in many graphical models, including finite state machines [McCallum *et al.*, 2000a; Lafferty *et al.*, 2001], where they are tied across different sequence indices, and by more complex patterns, as in Taskar *et al.* [2002]. Tied parameters use for extraction, classification, coreference and other relationships are described in section 3.4.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, SPAWARSYSCEN-SD grant numbers N66001-99-1-8912 and N66001-02-1-8903, Advanced Research and Development Activity under contract number MDA904-01-C-0984, and the Knowledge Discovery Laboratory and DARPA contract F30602-01-2-0566.

References

[Agrawal *et al.*, 1993] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[Anderson *et al.*, 2002] C. Anderson, P. Domingos, and D. Weld. Relational markov models and their application to adaptive web navigation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*. ACM Press, 2002.

[Appelt *et al.*, 1995] D. E. Appelt, Jerry R. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, and M. Tyson. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Morgan Kaufmann, 1995.

[Bikel *et al.*, 1997] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, pages 194–201, 1997.

[Bilenko and Mooney, 2002] Mikhail Bilenko and Raymond J. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical Report AI 02-296, Artificial Intelligence Lab, University of Texas at Austin, February 2002.

[Blei *et al.*, 2002] David Blei, Drew Bagnell, and Andrew McCallum. Learning with scope, with application to information extraction and classification. In *Uncertainty in Artificial Intelligence (UAI)*, 2002.

[Borthwick *et al.*, 2000] Andrew Borthwick, Vikki Papadouka, and Deborah Walker. The MEDD de-duplication project. In *Immunization Registry Conference*, 2000.

[Breiman *et al.*, 1984] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

[Breiman, 1996] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Bruninghaus and Ashley, 2001] Stefanie Bruninghaus and Kevin D. Ashley. Improving the representation of legal case texts with information extraction methods. In *Proceedings of the 8th international conference on Artificial Intelligence and Law*, 2001.

[Carreras *et al.*, 2002] Xavier Carreras, Lluis Marques, and Lluis Padro. Named entity extraction using adaboost. In *Proceedings of CoNLL-2002*, pages 167–170, 2002.

[Cohen and Hirsh, 1998] William W. Cohen and Haym Hirsh. Joins that generalize: text classification using WHIRL. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of KDD-98, 4th International Conference on Knowledge Discovery and Data Mining*, pages 169–173, New York, US, 1998. AAAI Press, Menlo Park, US.

[Collins, 2002] Michael Collins. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *ACL-02*, 2002.

[Craven *et al.*, 1998] M Craven, D DiPasquo, D Freitag, A McCallum, T Mitchell, K Nigam, and S Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 509–516, 1998.

[DARPA, 2002] DARPA. Darpa automatic content extraction program, 2002.

[Domingos and Richardson, 2001] P. Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 57–66, 2001.

[Fawcett and Provost, 1997] Tom Fawcett and Foster J. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[Freitag and McCallum, 1999] Dayne Freitag and Andrew Kachites McCallum. Information extraction with hmms and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Informatino Extraction*, 1999.

[Getoor *et al.*, 2001] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*. Springer-Verlag, 2001.

[Ghahramani and Jordan, 1995] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden Markov models. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, volume 8, pages 472–478. MIT Press, 1995.

[Ghani *et al.*, 2000] Rayid Ghani, Rosie Jones, Dunja Mladenic, Kamal Nigam, and Sean Slattery. Data mining on symbolic knowledge extracted from the web. In *KDD-2000 Workshop on Text Mining*, 2000.

[Hammersley and Clifford, 1971] J. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1971.

[Hearst, 1999] Marti Hearst. Untangling text data mining. In *Proceedings of ACL'99: the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.

[Jaakkola *et al.*, 2001] T. Jaakkola, M. Wainwright, and A. Willsky. Tree-based reparameterization for approximate estimation on graphs with cycles. In *Neural Information Processing Systems (NIPS)*, 2001.

[Jensen and Neville, 2002] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML2002)*, pages 259–266. Morgan Kaufmann, 2002.

[Jensen and Neville, 2003] D. Jensen and J. Neville. Randomization tests for relational learning. In *International Joint Conference on Artificial Intelligence (submitted)*, 2003.

[Jensen *et al.*, 2003a] D. Jensen, J. Neville, and M. Hay. Degree disparity leads to aggregation bias in relational models. In *International Conference on Machine Learning (submitted)*, 2003.

[Jensen *et al.*, 2003b] D. Jensen, M. Rattigan, and H. Blau. Misclassification errors and collective inference in relational data. In *Conference on Knowledge Discovery and Data Mining (submitted)*, 2003.

[Jensen, 1999] D. Jensen. Statistical challenges to inductive inference in linked data. In *Papers of the 7th International Workshop on Artificial Intelligence and Statistics*, 1999.

[Jerry *et al.*, 1996] H. Jerry, R. Douglas, E. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. Fastus: A cascaded finite-state transducer for extracting information from natural-language text, 1996.

[Jordan, 1998] M. Jordan, editor. *Learning in Graphical Models*. MIT Press, Cambridge, 1998.

[Klein *et al.*, 2003] Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher Manning. Named entity recognition with character-level models. In *Proceedings the Seventh Conference on Natural Language Learning*, 2003.

[Kosala and Blockeel, 2000] Kosala and Blockeel. Web mining research: A survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM*, 2, 2000.

[Lafferty *et al.*, 2001] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, pages 282–289, 2001.

[Lawrence *et al.*, 1999] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and Autonomous Citation Indexing. *IEEE Computer*, 32(6):67–71, 1999.

[Leek, 1997] Timothy R. Leek. Information extraction using hidden Markov models. Master's thesis, UC San Diego, 1997.

[Malouf, 2002] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Sixth Workshop on Computational Language Learning (CoNLL-2002)*, 2002.

[McCallum and Li, 2003] Andrew McCallum and Wei Li. Early results for named entity extraction with conditional random fields, feature induction and web-enhanced lexicons. In *Seventh Conference on Natural Language Learning (CoNLL)*, 2003.

[McCallum and Wellner, 2003] Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Information Integration on the Web*, 2003.

[McCallum *et al.*, 2000a] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of ICML*, pages 591–598, 2000.

[McCallum *et al.*, 2000b] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the contruction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000.

[McCallum *et al.*, 2000c] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining*, pages 169–178, 2000.

[McCallum, 2002] Andrew McCallum, 2002. Personal experience at WhizBang Labs, Inc.

[McCallum, 2003] Andrew McCallum. Efficiently inducing features of conditional random fields. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*, 2003.

[McCullagh and Nelder, 1989] P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman & Hall, New York, 1989.

[Miller *et al.*, 2000] Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. A novel use of statistical parsing to extract information from text. In *6th Applied Natural Language Processing Conference*, 2000.

[Mitchell, 1997] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[Morton, 1997] Thomas Morton. Coreference for NLP applications. In *Proceedings ACL*, 1997.

[Neville and Jensen, 2000] J. Neville and D. Jensen. Iterative classification in relational data. In L. Getoor and D. Jensen, editors, *Learning Statistical Models From Relational Data: Papers from the AAAI Workshop*, pages 42–49, Meno Park, CA, 2000. AAAI Press.

[Pasula *et al.*, 2002] Hanna Pasula, Bhaskara Marthi, Brian Milch, Stuart Russell, and Ilya Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing (NIPS)*, 2002.

[Pinto *et al.*, 2003] David Pinto, Andrew McCallum, Xen Lee, and W. Bruce Croft. Combining classifiers in text categorization. In *Submitted to SIGIR '03: Proceedings of the Twenty-sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.

[Quinlan, 1993] R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Ray and Craven, 2001] S. Ray and M. Craven. Representing sentence structure in hidden markov models for information extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 2001.

[Rohanimanesh and McCallum, 2003] Khashayar Rohanimanesh and Andrew McCallum. Factorial conditional random fields. Technical report, Department of Computer Science, University of Massachusetts, 2003.

[Roth and tau Yih, 2002] Dan Roth and Wen tau Yih. Probabilistic reasoning for entity and relation recognition. In *COLING'02*, 2002.

[Schapire, 1999] Robert E. Schapire. Theoretical views of boosting. *Lecture Notes in Computer Science*, 1572:1–10, 1999.

[Sha and Pereira, 2003a] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. Technical Report CIS TR MS-CIS-02-35, University of Pennsylvania, 2003.

[Sha and Pereira, 2003b] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology, NAACL*, 2003.

[Soderland, 1997] S. Soderland. Learning to extract text-based information from the world wide web. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, 1997.

[Taskar *et al.*, 2002] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*, 2002.

[Wang, 1999] Xue Z. Wang. *Data Mining and Knowledge Discovery for Process Monitoring and Control*. Springer Verlag, 1999.

[Zelenko *et al.*, 2003] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research (submitted)*, 2003.

The Variable Precision Rough Set Inductive Logic Programming Model — a Statistical Relational Learning Perspective

R. S. Milton[1], **V. Uma Maheswari**[2] **and Arul Siromoney**[2]

[1] Department of Computer Science, Madras Christian College, Chennai – 600 059, India
[2] School of Computer Science and Engineering, Anna University, Chennai – 600 025, India
Contact email: asiro@vsnl.com

Abstract

The Variable Precision Rough Set Inductive Logic Programming model (VPRSILP model) extends the Variable Precision Rough Set (VPRS) model to Inductive Logic Programming (ILP). The VPRSILP model is considered from the Statistical Relational Learning perspective, by comparing and contrasting it with Stochastic Logic Programs.

Keywords: Rough Set Theory; Variable Precision Rough Sets; Inductive Logic Programming; Machine Learning; Statistical Relational Learning

1 Introduction

Inductive Logic Programming (ILP) [Muggleton, 1991] is the research area formed at the intersection of logic programming and machine learning. ILP uses background knowledge, and positive and negative examples to induce a logic program that describes the examples. The induced logic program consists of the original background knowledge along with an induced hypothesis.

Rough set theory [Pawlak, 1982; 1991] defines an indiscernibility relation, where certain subsets of examples cannot be distinguished. A concept is rough when it contains at least one such indistinguishable subset that contains both positive and negative examples. It is inherently not possible to describe the examples accurately, since certain positive and negative examples cannot be distinguished.

The gRS–ILP model [Siromoney, 1997; Siromoney and Inoue, 2002] introduces a rough setting in Inductive Logic Programming. It describes the situation where the background knowledge, declarative bias and evidence are such that any induced logic program cannot distinguish between certain positive and negative examples. Any induced logic program will either cover both the positive and the negative examples in the group, or not cover the group at all, with both the positive and the negative examples in this group being left out.

The Variable Precision Rough Set (VPRS) model [Ziarko, 1993] is a generalized model of rough sets that inherits all basic mathematical properties of the original rough set model. Rough Set Theory assumes that the universe under consideration is known and all the conclusions derived from the model are applicable only to this universe. In practice, however, there is an evident need to generalize conclusions obtained from a smaller set of examples to a larger population. The VPRS model allows for a controlled degree of misclassification. Any partially incorrect classification rule provides valuable trend information about future test cases if the majority of available data to which such a rule applies can be correctly classified.

This paper presents the Variable Precision Rough Set Inductive Logic Programming model [Maheswari *et al.*, 2001b], an extension of the gRS–ILP model using features of the VPRS model, and compares and contrasts this model with Stochastic Logic Programs [Muggleton, 2000].

2 Inductive Logic Programming

The semantics of ILP systems are discussed in [Muggleton and Raedt, 1994]. In ILP systems, background (prior) knowledge B and evidence E (consisting of positive evidence E^+ and negative evidence E^-) are given, and the aim is then to find a hypothesis H such that certain conditions are fulfilled.

In the *normal semantics*, the background knowledge, evidence and hypothesis can be any well-formed logical formula. The conditions that are to be fulfilled by an ILP system in the normal semantics are

Prior Satisfiability: $B \wedge E^- \not\models \square$
Posterior Satisfiability: $B \wedge H \wedge E^- \not\models \square$
Prior Necessity: $B \not\models E^+$
Posterior Sufficiency: $B \wedge H \models E^+$

However, the *definite semantics*, which can be considered as a special case of the normal semantics, restricts the background knowledge and hypothesis to being definite clauses. This is simpler than the general setting of normal semantics, since a definite clause theory T has a unique minimal Herbrand model $\mathcal{M}^+(T)$, and any logical formula is either true or false in the minimal model. The conditions that are to be fulfilled by an ILP system in the definite semantics are

Prior Satisfiability: all $e \in E^-$ are false in $\mathcal{M}^+(B)$
Posterior Satisfiability: all $e \in E^-$ are false in $\mathcal{M}^+(B \wedge H)$
Prior Necessity: some $e \in E^+$ are false in $\mathcal{M}^+(B)$
Posterior Sufficiency: all $e \in E^+$ are true in $\mathcal{M}^+(B \wedge H)$

The Sufficiency criterion is also known as *completeness* with respect to positive evidence and the Posterior Satisfiability criterion is also known as *consistency* with the negative evidence.

The special case of definite semantics, where evidence is restricted to true and false ground facts (examples), is called the *example* setting. The example setting is thus the normal semantics with B and H as definite clauses and E as a set of ground unit clauses. The example setting is the main setting of ILP employed by the large majority of ILP systems.

3 Formal definitions of the gRS–ILP model

The generic Rough Set Inductive Logic Programming (gRS–ILP) model introduces the basic definition of elementary sets and a rough setting in ILP [Siromoney, 1997; Siromoney and Inoue, 2002]. The essential feature of an elementary set is that it consists of examples that cannot be distinguished from each other by any induced logic program in that ILP system. The essential feature of a rough setting is that it is inherently not possible for certain positive and negative examples to be distinguished, since both these positive and negative examples are in the same elementary set. The basic definitions formalised in [Siromoney and Inoue, 2000] follow.

The ILP system in the example setting of [Muggleton and Raedt, 1994] is formally defined as follows.

Definition 3.1. An *ILP system in the example setting* is a tuple $S_{es} = (E_{es}, B)$, where
(1) $E_{es} = E^+_{es} \cup E^-_{es}$ is the *universe*, where E^+_{es} is the set of positive examples (true ground facts), and E^-_{es} is the set of negative examples (false ground facts), and
(2) B is a background knowledge given as definite clauses such that (i) for all $e^- \in E^-_{es},\ B \not\vdash e^-$, and (ii) for some $e^+ \in E^+_{es},\ B \not\vdash e^+$.

Let $S_{es} = (E_{es}, B)$ be an ILP system in the example setting. Then let $\mathcal{H}(S_{es})$ (also written as $\mathcal{H}(E_{es}, B)$) denote the set of all possible definite clause hypotheses that can be induced from E_{es} and B, and be called the *hypothesis space* induced from S_{es} (or from E_{es} and B). Further, let $\mathcal{P}(S_{es})$ (also written as $\mathcal{P}(E_{es}, B) = \{P = B \wedge H \mid H \in \mathcal{H}(E_{es}, B)\}$) denote the set of all the programs induced from E_{es} and B, and be called the *program space* induced from S_{es} (or from E_{es} and B).

The aim is to find a program $P \in \mathcal{P}(S_{es})$ such that the next two conditions hold: (iii) for all $e^- \in E^-_{es},\ P \not\vdash e^-$, (iv) for all $e^+ \in E^+_{es},\ P \vdash e^+$.

The following simple illustration is used to explain this definition. Let $S = (E, B)$ where $E = E^+ \cup E^-$,
$E^+ = \{p(d1), p(d2), p(d3)\}$,
$E^- = \{p(d4), p(d5), p(d6)\}$ and
$B = \{atom(d1, c), atom(d2, c), atom(d3, o), atom(d4, o), atom(d5, n), atom(d6, n)\}$. Without loss of generality, only six examples are considered $p(d1)$, $p(d2)$, $p(d3)$, $p(d4)$, $p(d5)$, $p(d6)$ in our universe of examples. The background knowledge B indicates that the positive example molecule d_1 has a carbon atom, negative example molecule d_4 has an oxygen atom, negative example molecule d_5 has a nitrogen atom, and so on. The background knowledge B has only ground facts, using the predicate $atom$, and so does not cover any example. It is seen that for all $e^- \in E^-,\ B \not\vdash e^-$, and for some $e^+ \in E^+,\ B \not\vdash e^+$. (Two conditions (i) and (ii) of an ILP system in the example setting hold.) Let $H = \{p(d1), p(d2), p(d3)\}$. Then for all $e^- \in E^-,\ B \wedge H \not\vdash e^-$, and for all $e^+ \in E^+,\ B \wedge H \vdash e^+$. (Two conditions (iii) and (iv) also hold.)

The following definitions of Rough Set ILP systems in the gRS–ILP model (abbreviated as *RSILP systems*) use the terminology of [Muggleton and Raedt, 1994].

Definition 3.2. An *RSILP system in the example setting* (abbreviated as RSILP–E system) is an ILP system in the example setting, $S_{es} = (E_{es}, B)$, such that there does not exist a program $P \in \mathcal{P}(S_{es})$ satisfying both the conditions (iii) and (iv) above.

Definition 3.3. An *RSILP–E system in the single–predicate learning context* (abbreviated as RSILP–ES system) is an RSILP–E system, whose *universe* E is such that all examples (ground facts) in E use only one predicate, also known as the *target predicate*.

A *declarative bias* [Muggleton and Raedt, 1994] restricts the set of acceptable hypotheses, and is of two kinds: *syntactic bias* (also called *language bias*) that imposes restrictions on the form (syntax) of clauses allowed in the hypothesis, and *semantic bias* that imposes restrictions on the meaning, or the behaviour of hypotheses.

Definition 3.4. An *RSILP–ES system with declarative bias* (abbreviated as RSILP–ESD system) is a tuple $S = (S', L)$, where
(i) $S' = (E, B)$ is an RSILP–ES system, and
(ii) L is a declarative bias, which is any restriction imposed on the hypothesis space $\mathcal{H}(E, B)$.
We also write $S = (E, B, L)$ instead of $S = (S', L)$.

For any RSILP–ESD system $S = (E, B, L)$, let
$\mathcal{H}(S) = \{H \in \mathcal{H}(E, B) \mid H$ is allowed by $L\}$, and
$\mathcal{P}(S) = \{P = B \wedge H \mid H \in \mathcal{H}(S)\}$.
$\mathcal{H}(S)$ (also written as $\mathcal{H}(E, B, L)$) is called the *hypothesis space* induced from S (or from E, B, and L). $\mathcal{P}(S)$ (also written as $\mathcal{P}(E, B, L)$) denotes the set of all the programs induced by S, and is called the *program space* induced from S (or from E, B, and L).

It is seen in the illustration used earlier that the ILP system can exactly describe the set of positive examples, but in a manner that is not very useful, since the hypothesis is the same as the positive example ground facts. If the terms $d1, \ldots, d6$ are not allowed in H, then with $H = \{p(A) \leftarrow$

$atom(A, c)\}$, for all $e^- \in E^-$, $B \wedge H \not\vdash e^-$. However it is not true that for all $e^+ \in E^+$, $B \wedge H \vdash e^+$, since $B \wedge H \not\vdash p(d3) \in E^+$. (Condition (iii) holds, but not condition (iv).)
With $H = \{p(A) \leftarrow atom(A, c), p(A) \leftarrow atom(A, o)\}$, for all $e^+ \in E^+$, $B \wedge H \vdash e^+$. However it is not true that for all $e^- \in E^-$, $B \wedge H \not\vdash e^-$, since $B \wedge H \vdash p(d4) \in E^-$. (Condition (iv) holds, but not condition (iii).)

This is formalised in the definition of the RSILP–ESD system. Let $S = (E, B, L)$ where E and B are as given above, and L is the declarative bias such that $d1, \ldots, d6$ is not a term in $q(\ldots)$ for any $H \in \mathcal{H}(S)$, any $C \in H$, and any predicate $q(\ldots) \in C$.

An equivalence relation on the universe of an RSILP–ESD system is now defined.

Definition 3.5. Let $S = (E, B, L)$ be an RSILP–ESD system. An indiscernibility relation of S, denoted by $R(S)$, is a relation on E defined as follows: $\forall x, y \in E,\ (x, y) \in R(S)$ iff
$(P \vdash x \Leftrightarrow P \vdash y)$ for any $P \in \mathcal{P}(S)$ (i.e. iff x and y are inherently indistinguishable by any induced logic program P in $\mathcal{P}(S)$).

The following fact follows directly from the definition of $R(S)$.

Fact 1 *For any RSILP–ESD system S, $R(S)$ is an equivalence relation.*

Definition 3.6. Let $S = (E, B, L)$ be an RSILP–ESD system. An *elementary set* of $R(S)$ is an equivalence class of the relation $R(S)$. For each $x \in E$, let $[x]_{R(S)}$ denote the elementary set of $R(S)$ containing x. Formally,
$[x]_{R(S)} = \{y \in E \mid (x, y) \in R(S)\}$.
A *composed set* of $R(S)$ is any finite union of elementary sets of $R(S)$.

Definition 3.7. An RSILP–ESD system $S = (E, B, L)$ is said to be in a *rough setting* iff
$\exists e^+ \in E^+\ \exists e^- \in E^-\ (\ (e^+, e^-) \in R(S)\ )$.

It is seen from E, B, and L in the illustration used earlier that $R(S) = \{\ (p(d1), p(d2)),\ (p(d2), p(d1)),\ (p(d3), p(d4)),\ (p(d4), p(d3)),\ (p(d5), p(d6)),\ (p(d6), p(d5))\ \}$.
The elementary sets of $R(S)$ are
$\{p(d1), p(d2)\}, \{p(d3), p(d4)\}, \{p(d5), p(d6)\}$.
The composed sets of $R(S)$ are
$\{\}, \{p(d1), p(d2)\}, \ldots, \{p(d1), p(d2), p(d3), p(d4)\}, \ldots,$
$\{p(d1), p(d2), p(d3), p(d4), p(d5), p(d6)\}$.
S is in a rough setting since $p(d3) \in E^+$, $p(d4) \in E^-$ and $(p(d3), p(d4)) \in R(S)$.

Other work in Rough Set Inductive Logic Programming include [Midelfart and Komorowski, 2000; Liu and Zhong, 1999].

4 Formal definitions of the VPRSILP model

The formal definitions of the VPRSILP model are defined in [Maheswari *et al.*, 2001b].

A parameter β, a real number in the range $(0.5, 1]$, is used in the VPRS model as a threshold in elementary sets that have both positive and negative examples. This threshold is used to decide if that elementary set can be classified as positive or negative, depending on the statistical occurrence of positive and negative examples in it.

Definition 4.1. A *Variable Precision RSILP–ESD system* (abbreviated as VPRSILP–ESD system) is a tuple $S = (S', \beta)$, where
(i) $S' = (E, B, L)$ is an RSILP–ESD system, and
(ii) β is a real number in the range $(0.5, 1]$.
It is also written $S = (E, B, L, \beta)$ instead of $S = (S', \beta)$.

The definitions of hypothesis space, program space, equivalence relation, elementary sets, composed sets and rough setting defined above for RSILP–ESD systems hold for the VPRSILP–ESD system.

The following definitions use the VPRS terminology from [An *et al.*, 1997].

Definition 4.2. The *conditional probability* $P(E^+ \mid [x]_{R(S)})$ is defined as

$$P(E^+ \mid [x]_{R(S)}) = \frac{P(E^+ \cap [x]_{R(S)})}{P([x]_{R(S)})} = \frac{\mid E^+ \cap [x]_{R(S)}) \mid}{\mid [x]_{R(S)} \mid}$$

where $P(E^+ \mid [x]_{R(S)})$ is the probability of occurrence of event E^+ conditioned on event $[x]_{R(S)}$.

It is noted that $P(E^+ \mid [x]_{R(S)}) = 1$ if and only if $[x]_{R(S)} \subseteq E^+$;
$P(E^+ \mid [x]_{R(S)}) > 0$ if and only if $[x]_{R(S)} \cap E^+ \neq \emptyset$;
and $P(E^+ \mid [x]_{R(S)}) = 0$ if and only if $[x]_{R(S)} \cap E^+ = \emptyset$.

Definition 4.3. The *β–positive region* of S, $Pos_\beta(S)$, is defined as
$Pos_\beta(S) = \bigcup_{P(E^+ \mid [x]_{R(S)}) >= \beta, \text{ for all } [x]_{R(S)} \text{ in } R(S)} \{[x]_{R(S)}\}$
The *β–negative region* of S, $Neg_\beta(S)$, is defined as
$Neg_\beta(S) = \bigcup_{P(E^+ \mid [x]_{R(S)}) < \beta, \text{ for all } [x]_{R(S)} \text{ in } R(S)} \{[x]_{R(S)}\}$

Definition 4.4. The *β–restricted program space* of S, $\mathcal{P}_\beta(S)$ (also written as $\mathcal{P}_\beta(E, B, L, \beta)$), is defined as
$\mathcal{P}_\beta(S) = \{P \in \mathcal{P}(S) \mid P \vdash x \Rightarrow x \in Pos_\beta(S)\}$.
Any $P \in \mathcal{P}_\beta(S)$ is called a β–restricted program of S.

Our aim is to find a hypothesis H such that $P = B \wedge H \in \mathcal{P}_\beta(\mathcal{S})$.

The VPRSILP model has been applied in illustrative experiments to determine the transmembrane domains in amino

acid sequences [Maheswari *et al.*, 2001b] and to analyse and classify web log data [Maheswari *et al.*, 2001a; 2003].

5 The VPRSILP model and Stochastic Logic Programs (SLP)

A clause C is said to be *range–restricted* if and only if every variable in the head of C is found in the body of C. A *stochastic clause* is a pair $p : C$ where p in the interval $[0, 1]$ is the probability associated with C, a range–restricted clause. A set of stochastic clauses P is called a *Stochastic Logic Program* (SLP) if and only if for each predicate symbol q in P the probabilities associated with all clauses with q in the head sum to 1 [Muggleton, 2000]. Every example derived from an SLP has a probability associated with it. This is the product of the probabilities associated with every clause used in the derivation of the example.

In a VPRSILP–ESD system $S = (E, B, L, \beta)$, every example x in E falls into one of the elementary sets, $[x]_{R(S)}$. An elementary set in VPRSILP, by definition, consists of examples that are indistinguishable by any logic program P that can be induced from the examples E, background knowledge B and declarative bias L. Every elementary set $[x]_{R(S)}$ has a conditional probability $P(E^+ \mid [x]_{R(S)})$ associated with it depending on the statistical occurrence of positive and negative examples in it.

Hence every example in VPRSILP has a probability based value associated with it, namely the conditional probability of the elementary set in which this example occurs. This is in some sense similar to the probability associated with an example derived from a stochastic logic program.

However, further study of the comparison between these two models is required.

The following points are observed.

In VPRSILP, the conditional probability associated with an elementary set is based on the examples. In other words, the conditional probability is obtained from the observed data. In SLP, a probability is associated with each clause, probably using domain knowledge.

In VPRSILP, a probability value is associated with the negative examples also. In SLP a probability value of 0 is assigned to an example that is rejected by the SLP.

In VPRSILP, the conditional probability associated with an elementary set uses both negative and positive examples in that elementary set. In other words, the negative examples also play a role in determining the probability value. In SLP, the probability value is associated with a clause, and so only the examples that are derived (the positive examples) play a role.

In VPRSILP, the conditional probabilities associated with the elementary sets do not form a probability distribution. In SLP, the probabilities associated with each clause with the same head predicate form a probability distribution.

6 The VPRSILP model and application to Predictive Toxicology

In this section, the cVPRSILP approach based on the VPRSILP model is outlined [Milton *et al.*, 2003]. In the cVPRSILP approach, elementary sets are defined using attributes that are based on a finite number of clauses of interest.

Predictive Toxicology Evaluation

The rodent carcinogenicity tests conducted within the US National Toxicology Program by the National Institute of Environmental Health Sciences (NIEHS). has resulted in a large database of compounds classified as carcinogens or otherwise. The Predictive Toxicology Evaluation project of the NIEHS provided the opportunity to compare carcinogenicity predictions on previously untested chemicals. This presented a formidable challenge for programs concerned with knowledge discovery. The ILP system Progol [Muggleton, 1995] has been used in this Predictive Toxicology Evaluation Challenge [Srinivasan *et al.*, 1997a; 1997b].

Elementary Sets

In [Pawlak and Skowron, 1999], two finite, nonempty sets U and A are considered, where U is the universe of objects, and A is a set of attributes. With every attribute $a \in A$ is associated a set V_a of its values, called the domain of a.

The set of attributes A determines a binary relation R on U. R is an indiscernibility relation, defined as follows: xRy if and only if $a(x) = a(y)$ for every $a \in A$; where $a(x) \in V_a$ denotes the value of attribute a for object x. Obviously R is an equivalence relation. Equivalence classes of the relation R are referred to as elementary sets.

In cVPRSILP, let $A = \{A_1, \ldots, A_{i_{max}}\}$ be the set of attributes, with $V_a = \{\texttt{true}, \texttt{false}\}$ for every $a \in A$. Every $A_i \in A$ is associated with the clauses of interest $C'_i, i = 1, ..., i_{max}$, such that $A_i = \texttt{true}$ if the example can be derived from $C'_i \wedge B$, and $A_i = \texttt{false}$ otherwise. In this context, it is seen that these attributes form an equivalence relation.

Beta positive and beta negative regions

Elementary sets formed from the training examples fall into either the β–positive or the β–negative region, depending on the value of β.

A test example is decided as being positive or negative, depending on whether its elementary set is in the β–positive or the β–negative region.

Experimental illustration

An illustrative experiment is performed using the cVPRSILP model. The dataset used is the Predictive Toxicology Evaluation Challenge dataset found at
`http://web.comlab.ox.ac.uk/oucl/research/areas/`

In this experimental illustration, two predicates `has_property` and `atm` with four properties and three atom types are considered. These have been heuristically chosen based on visual inspection of clauses induced by Progol. Further studies are in progress to arrive at a more systematic choice.

The maximum number of predicates in a clause is taken as 2 and a finite set of clauses of interest is generated.

Each of the clauses of interest is treated as an attribute, and every example is placed in the appropriate elementary set, based on the subset of clauses which cover that example. Each elementary set falls in the β–positive or the β–negative region, depending on the chosen value of β. In this illustration, we use the value of 0.5. An example is predicted positive if its elementary set falls in the β–positive region, and is predicted negative if the elementary set falls in the β–negative region.

The following table is obtained when prediction is done on the training set itself. The overall prediction accuracy is 86%. Further analysis needs to be done.

	Actual Positive	Actual Negative	
Predicted Positive	142	22	164
Predicted Negative	16	111	127
Unclassified	0	2	2
	158	135	293

7 Conclusions

The VPRSILP model combines statistical and relational perspectives. The utility of the model has already been shown in classification experiments in computational biology and web mining. A brief discussion on the similarities and differences of the model with Stochastic Logic Programs, a Statistical Relational Learning paradigm, is presented.

References

[An *et al.*, 1997] A. An, C. Chan, N. Shan, N. Cercone, and W. Ziarko. Applying knowledge discovery to predict water-supply consumption. *IEEE Expert*, 12(4):72–78, 1997.

[Liu and Zhong, 1999] C. Liu and N. Zhong. Rough problem settings for Inductive Logic Programming. In N.Zhong and A.Skowron ad S.Ohsuga, editors, *New Directions in Rough Sets, Data Mining, and Granular–Soft Computing — 7th International Workshop, RSFDGrC'99*, Lecture Notes in Artificial Intelligence 1711, pages 168–177, Yamaguchi, Japan, November 1999. Springer.

[Maheswari *et al.*, 2001a] V. Uma Maheswari, Arul Siromoney, and K. M. Mehata. The Variable Precision Rough Set Inductive Logic Programming model and web usage graphs. In *New Frontiers in Artificial Intelligence — Joint JSAI 2001 Workshop Post–Proceedings*, volume 2253, pages 339–343. Lecture Notes in Computer Science, Springer, 2001.

[Maheswari *et al.*, 2001b] V. Uma Maheswari, Arul Siromoney, K. M. Mehata, and K. Inoue. The Variable Precision Rough Set Inductive Logic Programming Model and Strings. *Computational Intelligence*, 17(3):460–471, August 2001.

[Maheswari *et al.*, 2003] V. Uma Maheswari, Arul Siromoney, and K. M. Mehata. The Variable Precision Rough Set Inductive Logic Programming model and future test cases in web usage mining. In Masahiro Inuiguchi, Shusaku Tsumoto, and Shoji Hirano, editors, *Rough Set Theory and Granular Computing*. Physica–Verlag, 2003.

[Midelfart and Komorowski, 2000] H. Midelfart and J. Komorowski. A Rough Set approach to Inductive Logic Programming. In W. Ziarko and Y. Yao, editors, *Rough Sets and Current Trends in Computing — Second International Conference, RSCTC 2000*, Lecture Notes in Artificial Intelligence 2005, pages 190–198, Banff, Canada, October 2000. Springer.

[Milton *et al.*, 2003] R. S. Milton, V. Uma Maheswari, and Arul Siromoney. The variable precision rough set inductive logic programming model and predictive toxicology. 2003. Submitted.

[Muggleton and Raedt, 1994] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19(20):629–679, 1994.

[Muggleton, 1991] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[Muggleton, 1995] S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[Muggleton, 2000] S. Muggleton. Learning Stochastic Logic Programs. In *AAAI 2000 Workshop on Learning Statististical Models from Relational Data*, Austin, Texas, USA, July 2000.

[Pawlak and Skowron, 1999] Z. Pawlak and A. Skowron. Rough set rudiments. In *Bulletin of International Rough Set Society*, volume 3. 1999.

[Pawlak, 1982] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

[Pawlak, 1991] Z. Pawlak. *Rough Sets — Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.

[Siromoney and Inoue, 2000] A. Siromoney and K. Inoue. Elementary sets and declarative biases in a restricted gRS–ILP model. *Informatica*, 24:125–135, 2000.

[Siromoney and Inoue, 2002] A. Siromoney and K. Inoue. The generic Rough Set Inductive Logic Programming (gRS–ILP) model. In T. Y. Lin, Y. Y. Yao, and L. A. Zadeh, editors, *Data Mining, Rough Sets and Granular Computing*, volume 95, pages 499–517. Physica–Verlag, 2002.

[Siromoney, 1997] A. Siromoney. A rough set perspective of Inductive Logic Programming. In Luc De Raedt and Stephen Muggleton, editors, *Proceedings of the IJCAI-97 Workshop on Frontiers of Inductive Logic Programming*, pages 111–113, Nagoya, Japan, 1997.

[Srinivasan *et al.*, 1997a] A. Srinivasan, R.D. King, S.H. Muggleton, and M. Sternberg. The predictive toxicology evaluation challenge. In *Proceedings of the Fifteenth International Joint Conference Artificial Intelligence (IJCAI-97)*, pages 1–6. Morgan-Kaufmann, 1997.

[Srinivasan *et al.*, 1997b] A. Srinivasan, R.D. King, S.H. Muggleton, and M. Sternberg. Carcinogenesis predictions using ILP. In N. Lavrač and S. Džeroski, editors, *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, pages 273–287. Springer-Verlag, Berlin, 1997. LNAI 1297.

[Ziarko, 1993] W. Ziarko. Variable precision rough set model. *Journal of Computer and System Sciences*, 46(1):39–59, 1993.

Statistical Relational Learning: Four Claims and a Survey

Jennifer Neville, Matthew Rattigan, David Jensen
Knowledge Discovery Laboratory, Department of Computer Science, University of Massachusetts,
140 Governors Drive, Amherst, MA 01003 USA
{jneville | rattigan | jensen } @cs.umass.edu

Statistical relational learning (SRL) research has made significant progress over the last 5 years. We have successfully demonstrated the feasibility of a number of probabilistic models for relational data, including probabilistic relational models, Bayesian logic programs, and relational probability trees, and the interest in SRL is growing. However, in order to sustain and nurture the growth of SRL as a subfield we need to refocus our efforts on the science of machine learning — moving from demonstrations to comparative and ablation studies. We will outline four assertions that are implicit to SRL research but which have been only minimally evaluated. We hope to stimulate discussion as to how, as a community, these claims can be addressed in future research.

1 Introduction

In the hopes of generalizing the results of recent research from the statistical relational learning (SRL) community, we surveyed twenty recent SRL papers. From the papers studied we were able to distill four implicit claims that underlie much of the current SRL research. We present an examination of those claims in the context of the papers surveyed.

We chose twelve of the papers as a representative sample for the purposes of this discussion. Each paper chosen describes and evaluates a discriminative, probabilistic relational model. A descriptive list of the selected models and papers appears in Table 1.

The purpose of this paper is to stimulate a discussion of the scientific methods that will help to illustrate and evaluate the relative merits of the different models and their frameworks.

2 Relational vs. propositional

Claim: Models learned from both intrinsic and relational information perform better than those learned from intrinsic information alone, and are therefore worth the added complexity.

This is an implicit claim of relational learning in general. We expect that predictive information exists in relationships among instances, and that this information can be used to reduce model bias. However, decreasing bias often results in increased variance (Friedman 1997). This is a very real concern for relational learning algorithms that are faced with an exponential explosion in the size of the model space.

The simplest way to evaluate this claim is to record model performance using *intrinsic* data, a subset of the data where relational information is removed. By this we mean data where the instances are objects in isolation, and the only information available are the attributes intrinsic to those objects as individuals. Popescul, Ungar, Lawrence, and Pennock (2003) use this approach when evaluating their models on citation data, comparing models learned on information intrinsic to documents alone with those learned from both intrinsic and citation information. Getoor, Segal, Taskar and Koller (2001) use an alternative approach, including results from a baseline propositional model learned on intrinsic data. This technique is also employed in four other papers. See figure 2 for details.

More than half of the papers surveyed included some comparative intrinsic analysis, and the results vary considerably across models and datasets. For example, when using relational features Neville, Jensen, Gallagher, and Fairgrieve (2003) found marked improvement in model performance on two datasets, but no significant gain on a third. We believe that this type of analysis is important baseline for determining whether the inclusion of relational information is of any benefit, and if so whether the additional model complexity is warranted. Although preliminary analysis along these lines is a common component of SRL research, we feel that more explicit and directed experimentation is needed to fully justify the use SRL models for relational datasets.

3 Probabilistic vs. deterministic

Claim: Probabilistic relational models offer significant advantages over deterministic relational models in relational domains.

Table 1: Statistical relational learning models surveyed

Model	Description	Selective	Generative	Reference
RVS	relational vector-space model	No	no	Bernstein, Clearwater, and Provost, 2003
FOIL-PILFS	relational learner w/statistical predicate invention	Yes	no	Craven and Slattery, 2001
Maccent	maximum entropy model with clausal constraints	Yes	no	Dehaspe, 1997
SNM	Markov random field for social networks	No	no	Domingos and Richardson, 2001
BLP	Bayesian logic programs	yes	yes	Kersting and De Raedt, 2002
1BC2	first-order naive Bayesian classifier	no	no	Lachiche and Flach, 2002
RBC	relational Bayes classifier	no	no	Neville, Jensen, Gallagher and Fairgrieve, 2003
RPT	relational probability trees	yes	no	Neville, Jensen, Friedland and Hay, 2003
SLR	structural logistic regression	yes	no	Popescul, Ungar, Lawrence, and Pennock, 2003
NBILP-R	naive Bayes classifier with ILP features	no	no	Pompe and Kononenko, 1995
PRM	probabilistic relational model	yes	yes	Getoor, Segal, Taskar and Koller, 2001
RMN	relational Markov network	no	no	Taskar, Abbeel and Koller, 2002

Research in relational learning has investigated deterministic models for many years (e.g. Muggleton & De Raedt 1994, Lavrac & Dzeroski 1994). Recent efforts have shifted the focus towards a probabilistic setting. We outline a number of advantages of probabilistic models below, but we feel that discussion of the strengths and weaknesses of each technique is worth exploring in greater detail. Discussion along these lines is necessary to come to a general understanding of the range and applicability of SRL models.

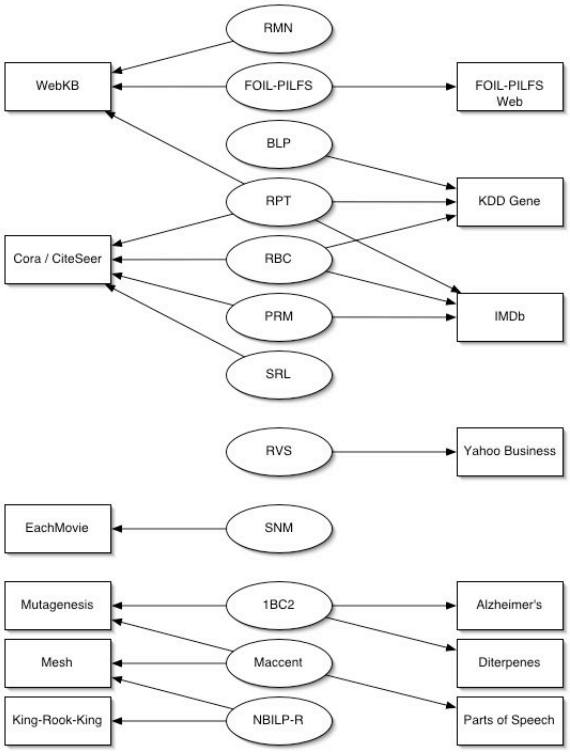


Figure 1: SRL models and evaluation datasets.

One strength of probabilistic models is the ability to evaluate how these models will perform over a range of class and cost distributions (Provost and Fawcett, 1997). Classification tasks involving complex relational data often have varying levels of misclassification costs as well as uncertain class distributions. Since deterministic models do not associate a level of confidence with their classifications, it is difficult to estimate their behavior in these domains.

Another advantage of probabilistic models is their suitability to real-world analysis tasks. Since these models generate meaningful, continuous probability scores, they lend themselves to an iterative, hierarchical approach to analysis. As Bernstein, Clearwater, and Provost (2003) point out, "scores may be most useful as feature constructors in other, more complicated systems." It is therefore crucial to evaluate the probabilities produced in SRL models quantitatively; unfortunately, none of the papers we surveyed perform this type of evaluation. Secondly, probability scores allow us to rank instances in order of certainty. This is of great use to real-world analysts who have limited time to investigate "positive" instances, as confidence scores allow an analyst to prioritize instances rather than treat all members of a predicted class equally.

Finally, probabilistic models are in general more suited to learning with relational data than deterministic ones. Due to their complexity, relational datasets are often noisy, which can be troublesome for deterministic models (Popescul et al. 2003). Furthermore, the advantage of working with relational data may be lost without the use of probabilistic models. For example, Craven and Slattery (2001) found in the text classification domain that "learned rules will not be dependent on the presence or absence of specific key words as a conventional relational method. Instead, the statistical classifiers in its learned rules consider the weighted evidence of many words."

4 Heterogeneous data

Claim: SRL algorithms learn accurate models of structured data.

Most conventional classification techniques assume data instances are recorded in homogeneous structures. Relational data however, often have complex structures that are difficult to model in propositional form. For example, information about actors, directors and producers may be useful when building a model of movie success but each movie has a different number of related entities. This variety results in examples with diverse structure — some movies may have 10 actors, and others may have hundreds. The ability to deal with heterogeneous data instances is a defining characteristic of relational learning algorithms.

The relational learning community has developed a number of models that can handle heterogeneous data. For example, Lachiche and Flach (2002) extend conventional naive Bayes classifiers to handle heterogeneous instances and Deshape (1997) extends conventional maximum entropy models to use a richer first-order logic format.

Each of the 12 papers surveyed introduces a different model for this purpose. However, few of these papers evaluate the effects of heterogeneity on the learned models. Some of our recent work has examined how particular characteristics of relational data affect the statistical inferences necessary for accurate learning (Jensen & Neville 2002, Jensen, Neville & Hay 2003). Specifically, we have shown that concentrated linkage combined with high autocorrelation can lead to feature selection bias if models are constructed naively. Also, we have shown that degree disparity can lead to spurious correlations in aggregated features, resulting in overly complex models with excess structures.

These characteristics of relational data can greatly complicate efforts to construct good statistical models. Only selective models are vulnerable to the particular biases mentioned above, but 7 of the models surveyed do some form of selection while learning. It is difficult to evaluate models for unidentified biases; however, comparative studies among the various SRL algorithms should help to uncover these biases. In particular, detailed comparisons of selective and non-selective model performance may help to uncover additional biases. Figure 2 depicts the 12 SRL models with links to the various models compared to during evaluation. The paucity of outlinks speaks for itself.

We have only begun to explore the effects of data characteristics on model learning. While many relational models outperform propositional models on the same datasets, the relational models may not be living up to their full potential. Further investigation of the complexities of relational data will help to identify sources of potential bias and correcting for these biases will unleash the full power of SRL models.

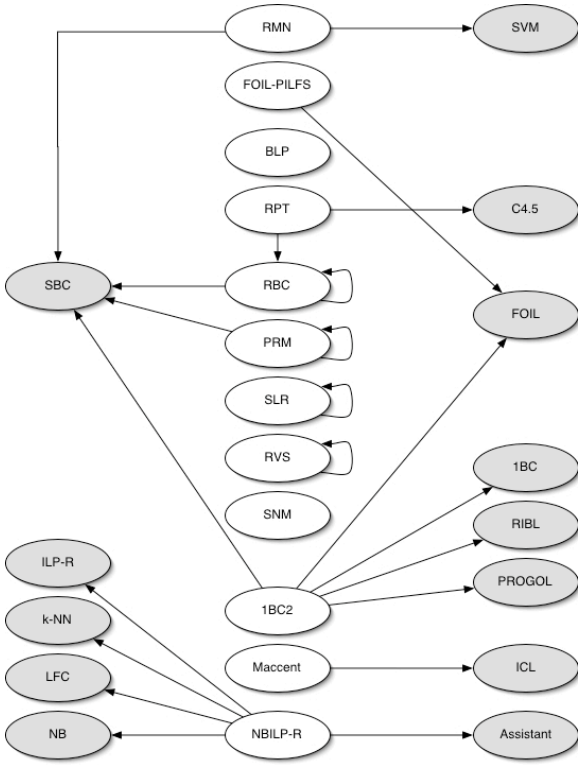


Figure 2: SRL models and evaluation models. Self-loops indicate ablation comparisons.

5 Interdependent data

Claim: SRL algorithms learn accurate models of dependent data instances.

Independence of instances is a deeply buried assumption of traditional machine learning methods that is contradicted by many relational datasets. For example, in scientific literature datasets there are dependencies among papers written by the same author and in web datasets there are dependencies among pages linked to by the same document. The structure of complex relational data such as these presents a unique opportunity for improving the accuracy of statistical models. If two objects are related, inferring something about one object can aid inferences about the other.

In our analysis of relational data, we have encountered many examples of dependencies that could be exploited to improve learning. For example, in analysis of the 2001 KDD Cup data we found that the proteins located in the same place in a cell (e.g., mitochondria or cell wall) had highly autocorrelated functions (e.g., transcription or cell growth). Such autocorrelation has been identified in other domains as well. For example, fraud in mobile phone networks has been found to be highly autocorrelated (Cortes, Pregibon & Volinsky 2001). The topics of authoritative web pages are highly autocorrelated when linked through directory pages that serve as "hubs" (Kleinberg 2001).

Table 2: Characteristics of data and sampling approach used for evaluation

Model	Data connectivity	Sampling approach
RVS	one large connected component	not mentioned
FOIL-PILFS	disjoint large graphs	leave-one-graph-out cross validation
Maccent	disjoint small graphs	leave-k-graph-out cross validation
SNM	one large connected component	sample by time
BLP	one large connected component	transduction
1BC2	disjoint small graphs	leave-k-graph-out cross validation
RBC	one large connected component	subgraph sampling
RPT	one large connected component	subgraph sampling
SLR	one large connected component	transduction
NBILP-R	disjoint small graphs	leave-k-graph-out cross validation
PRM	large conn comp / disjoint graphs	transduction / leave-one-graph-out cv
RMN	disjoint large graphs	leave-one-graph-out cross validation

Many of the models surveyed do not attempt to exploit dependencies among relational instances. More than half of the algorithms are designed to learn models relational datasets with independent, heterogeneous instances (i.i.d. relational data) where any dependencies among instances are ignored.

Inductive logic programming (ILP) models have been capable of representing dependencies among instances for years, albeit only extreme (deterministic) dependencies (Lavrac & Dzeroski 1994). However, it is only recently that statistical models have been developed to exploit the dependencies in relational data. For example, Kersting and De Raedt (2002) combine ILP with Bayesian networks to integrate probabilities into logic programs and model the dependencies among proteins in a cell. Getoor et al. (2001) use probabilistic relational models (PRMs) to model the the dependencies among hyperlinked web pages. Taskar, Abbeel and Koller (2002) use conditional Markov networks to model the same domain. Domingos and Richardson (2001) represent market entities as social networks and develop Markov random field models to model the influence in purchasing patterns throughout the network. Bernstein, Clearwater and Provost (2003) outline a relational vector-space model that uses autocorrelation to identify the group membership of linked entities.

Statistical models capable of collective classification across a network of instances are a relatively new phenomenon. It is unclear how to effectively evaluate the performance of these models. In what context do we expect to be using these models in the real world? Will we be applying the model to a completely new graph or do we expect new instances to arrive temporally related to the existing (training set) graph. Answers to this question should help to develop sampling methods to get an unbiased estimate of model performance.

Furthermore, how should we sample from a large connected graph? Table 2 outlines the characteristics of datasets examined by each of the models along with the sampling approach that was chosen. There are four approaches to sampling currently in use; more work is needed to determine which of these approaches is appropriate for a particular learning task.

6 Conclusions

Although the SRL community has successfully demonstrated the feasibility of a number of probabilistic models for relational data, there is much work to be done in order to begin generalizing the range and applicability of the various models. We have presented four claims for discussion with the purpose of advancing the science of SRL as well as machine learning in general.

References

Bernstein, A., S. Clearwater, and F. Provost. The relational vector-space model and industry classification. CDeR working paper #IS-03-02, Stern School of Business, New York University, 2003.

Cortes, C., D. Pregibon, and C. Volinsky. Communities of Interest. *Proceedings of the Fourth International Symposium on Intelligent Data Analysis*, 2001.

Craven, M. and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning Journal* 43:97-119, 2001.

Dehaspe, L. Maximum entropy modeling with clausal constraints. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, pages 109-124. Springer-Verlag, 1997.

Domingos, P., M. Richardson. Mining the Network Value of Customers. *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pp. 57-66, 2001.

Friedman, J. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55-77, 1997.

Getoor, L., E. Segal, B. Taskar and D. Koller. Probabilistic Models of Text and Link Structure for Hypertext Classification. *Proceedings of the IJCAI01 Workshop on Text Learning: Beyond Supervision*, 2001.

Jensen, D., J. Neville and M. Hay. Avoiding bias when aggregating relational data with degree disparity. *Proceedings of the 20th Int. Joint Conf. on Machine Learning*, to appear.

Kersting, K., L. De Raedt. Basic principles of learning Bayesian logic programs. Technical Report No. 174, Institute for Computer Science, University of Freiburg, Germany, June 2002.

Kleinberg, J. Authoritative sources in a hyperlinked environment. *Journal of the ACM* 46:604-632, 1999

Lachiche, N. and P. Flach. 1BC2: a True First-Order Bayesian Classifier. *Proceedings of the 12th International Conference on Inductive Logic Programming*, 2002.

Lavrac, N. and Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.

Muggleton, S. editor. *Inductive Logic Programming*. Academic Press, 1992.

Neville, J., D. Jensen, B. Gallagher, R. Fairgrieve. Simple Estimators for Relational Data. University of Massachusetts, Technical Report 03-04, 2003.

Neville, J., D. Jensen, L. Friedland, M. Hay. Learning relational probability trees. *Proceedings of the 9th International Conference on Knowledge Discovery & Data Mining,* to appear.

Popescul, A., L. Ungar, S. Lawrence, D. Pennock, Statistical relational learning for document mining. Submitted, 2003.

Pompe, U. and I. Kononenko. Naive Bayesian classifier within ILP-R. *Proceedings of the 5th International Workshop on Inductive Logic Programming,* pages 417-436, 1995.

Provost, F. and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. *Proceedings of the 3rd International Conference on Knowledge Discovery & Data Mining,* pages 43-48, 1997.

Taskar, B., P. Abbeel and D. Koller. Discriminative probabilistic models for relational data. *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 2002.

Parameter Estimation for Stochastic Context-Free Graph Grammars

Tim Oates, Shailesh Doshi, and Fang Huang
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County, Baltimore, MD 21250
{oates,sdoshi1,fhuang2}@cs.umbc.edu

Abstract

Given a sample from an unknown probability distribution over strings, there exist algorithms for inferring the structure and parameters of stochastic grammatical representations of the unknown distribution, i.e. string grammars. Despite the fact that research on grammatical representations of sets of graphs has been conducted since the late 1960's, almost no work has considered the possibility of stochastic graph grammars and no algorithms exist for inferring stochastic graph grammars from data. This paper presents PEGG, an algorithm for estimating the parameters of stochastic context-free graph grammars given a sample from an unknown probability distribution over graphs. It is established formally that PEGG finds parameter estimates in polynomial time that maximize the likelihood of the data, and preliminary empirical results demonstrate that the algorithm performs well in practice.

1 Introduction

Graphs are a natural representation for relational data. Nodes correspond to entities, edges correspond to relations, and symbolic or numeric labels on nodes and edges provide additional information about particular entities and relations. Graphs are routinely used to represent everything from social networks [Pattison, 1993] to chemical compounds [Cook et al., 1994] to visual scenes [Hong & Huang, 2002].

Suppose you have a set of graphs, each representing an observed instance of a known money laundering scheme [Office of Technology Assessment, 1995]. It would be useful to learn a statistical model of these graphs that supports the following operations:

- *Compute graph probabilities:* If the model represents a probability distribution over graphs, then it is possible to determine the probability of a new graph given the model. In the context of money laundering schemes, this would amount to determining whether a newly observed set of business relationships and transactions (represented as a graph) is likely to be an instance of money laundering.
- *Identify recurring structures:* Money laundering schemes may contain common components (i.e. subgraphs) that are arranged in a variety of ways. To better understand the domain, it is useful to explicitly identify such components and the common ways in which they are connected to one another.
- *Sample new graphs:* Given the model, one might want to sample new graphs (money laundering schemes) according to the probability distribution defined by the model. This might be useful in exploring the space of possible schemes, perhaps looking for new variants that law enforcement has not previously considered, or for generating training examples from which humans or programs can learn.

Stochastic grammatical representations of probability distributions over strings, such as stochastic context-free grammars (SCFGs), support these three operations. Given a SCFG, G, and a string, s, it is possible to efficiently compute $p(s|G)$. It is also trivial to sample strings from the probability distribution defined by G. Finally, there exist a number of methods for learning both the structure [Stolcke, 1994] and parameters [Lari & Young, 1990] of string grammars from data. The most well-known algorithm for computing maximum likelihood estimates of the parameters of string grammars is the Inside-Outside algorithm. In addition to estimating parameters, this algorithm can be used to learn structure. This is done by constructing a grammar containing, for example, all possible CNF productions that can be created from a given set of terminals and nonterminals. Inside-Outside can then prune away (i.e. set production probabilities to zero) those productions that are possible but that are not actually in the grammar that generated the training data. Also, Inside-Outside can be used as a component in a system that explicitly searches over the space of grammar structures, iteratively evaluating structures/parameters via, for example, the description length of the grammar and the data given the grammar.

We have embarked on a program of research aimed at creating algorithms for learning and reasoning with

stochastic grammatical representations of probability distributions over graphs that provide functionality mirroring that available for string grammars. There exists a fairly extensive literature on deterministic graph grammars that define sets of graphs in the language of the grammar (see, for example, [Engelfriet & Rozenberg, 1997] and [Ehrig et al., 1999]), just as deterministic string grammars define sets of strings that are in the language of the grammar. However, the vast majority of existing work on graph grammars has completely ignored the possibility of stochastic graph grammars and there is no work whatsoever on learning either the structure or parameters of graph grammars.

This paper describes an algorithm for estimating the parameters of stochastic graph grammars that we call Parameter Estimation for Graph Grammars (PEGG), the first algorithm of its kind. PEGG is similar in many respects to the Inside-Outside algorithm. PEGG computes inside and outside probabilities in polynomial time, and can use these probabilities to efficiently compute $p(g|G)$, the probability of graph g given graph grammar G. In addition, PEGG computes maximum likelihood estimates of grammar parameters for a given grammar structure and set of graphs, again in polynomial time. Though we have explored the use of Bayesian model merging techniques developed for learning the structure (i.e. productions) of string grammars [Stolcke, 1994] in the context of learning the structure of graph grammars [Doshi et al., 2002], the current focus is on parameter estimation.

The ability to learn grammar-based representations of probability distributions over graphs has the attractive property that non-terminals encode information about classes of functionally equivalent sub-graphs. For example, most money laundering schemes have a method for introducing illegal funds into the financial system and a method for moving the funds around to distance them from the source. If sub-graphs in the ground instances of money laundering schemes correspond to these methods, and there are different instantiations of each method, it is reasonable to expect that the learned grammar will contain a non-terminal that expands to ways of introducing funds into the financial system and another non-terminal that expands to ways of moving these funds around. Identifying these non-terminals in the learned grammar makes it possible to enumerate the sub-graphs they generate (i.e. all possible instantiations of a method) and to determine their probability of occurrence to, for example, focus law enforcement efforts.

From a more formal standpoint, graphs are logical structures, so individual graphs and sets of graphs can can be described by logical formulas. It is possible to deduce properties of graphs and sets of graphs from these descriptions [Immerman, 1999]. PEGG opens up the possibility of automatically synthesizing logical descriptions (i.e. graph grammars) of sets of graphs from data. For example, the expressive power of certain graph grammar formalisms is co-extensive with that of monadic second-order logic [Courcelle, 1997].

The remainder of this paper is organized as follows. Section 2 describes stochastic context-free graph grammars and discusses their relationship to stochastic context-free string grammars. Despite the fact that graph grammars have a rich history of application in a variety of domains, no algorithms exist for learning them from data. To introduce the fundamental concepts of grammar induction from data, section 3 reviews the Inside-Outside algorithm for estimating the parameters of stochastic context-free string grammars. Section 4 introduces the Parameter Estimation for Graph Grammars (PEGG) algorithm for learning maximum likelihood parameter estimates for graph grammars. Section 5 presents the results of a set of preliminary experiments with PEGG. Section 6 reviews related work, concludes, and discusses a number of directions in which we are taking this research.

2 Graph Grammars

This section provides an overview of graph grammars. For a thorough introduction to the formal foundations of graph grammars see [Engelfriet & Rozenberg, 1997], and to learn more about the vast array of domains in which graph grammars have been applied, see [Ehrig et al., 1999].

The easiest way to build intuition about graph grammars is by way of comparison with string grammars, for which we will take stochastic context-free grammars to be the paradigmatic example. (For the remainder of this paper the term *string grammar* means stochastic context-free string grammar.) Recall that a string grammar G is a 4-tuple (S, N, Σ, P) where N is a set of non-terminal symbols, $S \in N$ is the start symbol, Σ is a set of terminal symbols disjoint from N, and P is a set of productions. Associated with each production is a probability such that the probabilities for productions with the same left-hand side sum to one. Sometimes it will be convenient to describe grammars as being composed of *structure* and *parameters*, where the parameters are the production probabilities and the structure is everything else.

In this paper we will be concerned exclusively with stochastic context-free graph grammars [Mosbah, 1994], and will use the term *graph grammar* to refer to grammars of this type. Despite the fact that our present concern is with stochastic graph grammars, it is important to note that prior work reported in the literature has focused almost exclusively on deterministic grammars.

Just as string grammars define probability distributions over strings, graph grammars define probability distributions over graphs. A graph grammar G is a 4-tuple (S, N, Σ, P) where N is a set of non-terminal symbols, $S \in N$ is the start symbol, Σ is a set of terminal symbols disjoint from N, and P is a set of productions. Associated with each production is a probability such that the probabilities for productions with the same left-hand side sum to one.

The primary difference between string grammars and

graph grammars lies in the right-hand sides of productions. String grammar productions have strings of terminals and non-terminals on their right-hand sides. Graph grammar productions have graphs on their right-hand sides. At this point the reader may well be wondering where the terminals and non-terminals appear in the graphs generated by graph grammars. It turns out that they can be associated with nodes, yielding a class of grammars known as Node Controlled Embedding (NCE) graph grammars, or they can be associated with edges, yielding a class of grammars known as Hyperedge Replacement (HR) graph grammars. For reasons that will be discussed later, we focus exclusively on HR grammars.

Figure 1 shows the three productions in a simple HR grammar [Drewes et al., 1997] that has one non-terminal - S. Each left-hand side is a single non-terminal and each right-hand side is a graph. Some of the edges in the graphs are labeled with non-terminals in boxes. These *non-terminal edges* can be expanded, a process that involves removing the edge and replacing it with the graph on the right-hand side of a matching production. Each right-hand side has a pair of nodes labeled 1 and 2 that are used to orient the graph when it replaces a non-terminal edge. We will generally use the term *host graph* to refer to the graph containing the non-terminal edge and the term *sub-graph* to refer to the graph that replaces the non-terminal edge.

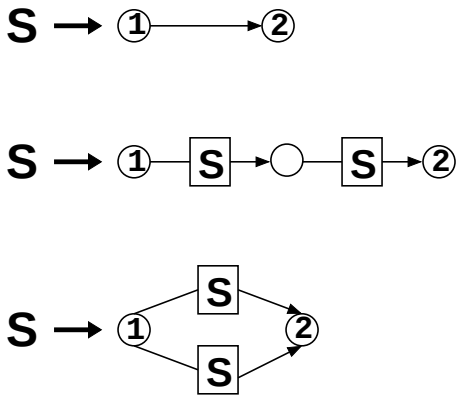


Figure 1: Productions in a simple HR grammar.

Figure 2 shows a partial derivation using the productions in figure 1. The second graph in figure 2 is obtained from the first by removing the labeled edge and replacing it with the sub-graph on the right-hand side of the second production in figure 1. After removing the edge, all that remains is two disconnected nodes, one that used to be at the head of the edge and the other at the tail. The edge is replaced by *gluing* the node labeled 1 in the sub-graph to the node that was at the head of the removed edge. Likewise, the node labeled 2 in the sub-graph is glued to (i.e. made the same node as) the node that was at the tail of the removed edge.

The last graph in figure 2 is obtained from the penultimate graph by replacing a non-terminal edge with the right-hand side of the first production in figure 1. This results in an edge with no label – a terminal edge – which can therefore not be expanded. A terminal graph is one that contains only terminal edges. Terminal edges can be unlabeled, as in the current example, or productions can specify labels for them from the set of non-terminals Σ.

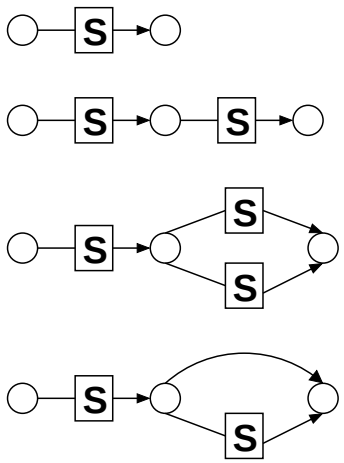


Figure 2: A partial derivation using the productions in figure 1.

Note that every production in figure 1 has exactly two distinguished nodes, labeled 1 and 2, that are used to orient the sub-graph in the host graph when an edge is replaced. When expanding a non-terminal in the derivation of a string there is no ambiguity about how to join the substrings to the left and right of the non-terminal with its expansion. Things are not so clear when expanding non-terminal edges to graphs. Given that the sub-graph to which the non-terminal is expanded will be attached by gluing, there are in general several possible attachments. Consider the second production in figure 1, whose right-hand side has three nodes. When it is used to replace a non-terminal edge, there are 6 possible ways of gluing the sub-graph to the host graph. Any of the three sub-graph nodes can be glued to the host graph node that was at the head of the non-terminal edge, and any of the remaining two sub-graph nodes can be glued to the host-graph node that was at the tail of the non-terminal edge. To remove this ambiguity, each production specifies which nodes in the sub-graph are to be glued to which nodes in the host graph.

In general, non-terminal edges can be *hyperedges* that join more than two nodes. A hyperedge is said to be an n-edge if it joins n nodes. All of the hyperedges in the above example are 2-edges, or simple edges. If an n-edge labeled with non-terminal X is to be expanded, there must be a production with X as its left-hand side and a graph on its right-hand side that has n distinguished nodes (e.g. labeled 1 - n) that will be glued to

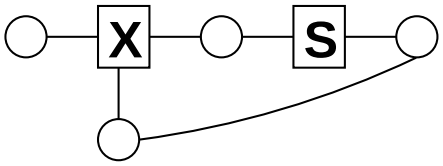


Figure 3: A graph containing a 3-edge labeled X.

the nodes that were attached to the hyperedge before it was removed. Figure 3 shows a graph containing an undirected 3-edge labeled X.

3 Parameter Estimation for String Grammars

Our goal is to develop a set of algorithms for graph grammars that mirror those available for string grammars, with the starting point being an algorithm for estimating the parameters of graph grammars from data. This section reviews the most widely used algorithm for estimating the parameters of string grammars from data - the Inside-Outside (IO) algorithm [Lari & Young, 1990]. This review will provide the necessary background for readers unfamiliar with IO and will make it possible to focus on issues specific to graph grammars in section 4 where we derive a version of IO for graph grammars (the PEGG algorithm).

Let $G = (\mathcal{S}, \theta)$ be a stochastic context-free string grammar with structure $\mathcal{S}$ and parameters θ. Let E be a set of training examples created by sampling from the probability distribution over strings defined by G. Given $\mathcal{S}$ and E, the goal of parameter estimation is to obtain a set of parameters, $\hat{\theta}$, such that $p(E|S, \hat{\theta})$ is maximized.

If G is unambiguous then maximum likelihood parameter estimation is easy. A grammar is *unambiguous* if every string in $L(G)$ has exactly one derivation [Hopcroft & Ullman, 1979]. That is, given a string in $L(G)$ it is possible to determine which productions were used to derive the string. Let $c(X \to \gamma|s)$ be the number of times production $X \to \gamma$ is used in the derivation of string s. Let $c(X \to \gamma|E)$ be $\sum_{s \in E} c(X \to \gamma|s)$. Then the maximum likelihood estimate for $p(X \to \gamma)$ is:

$$\hat{p}(X \to \gamma) = \frac{c(X \to \gamma|E)}{\sum_\delta c(X \to \delta|E)}$$

The estimate is simply the number of times X was expanded to γ divided by the number of times X occurred.

If G is ambiguous then strings in $L(G)$ can have multiple derivations, and parameter estimation becomes more difficult. The problem is that only the strings in E are observable, not their derivations. Given a string $s \in E$, one of the possibly many derivations of s was actually used to generate the string when sampling from the probability distribution over strings defined by G. It is production counts from this derivation, and none of the other legal derivations, that are needed for parameter estimation.

This is an example of a hidden data problem. Given information about which derivation was used when sampling each $s \in E$, the estimation problem is easy, but we do not have this information. As is typical in such cases the solution is Expectation Maximization (EM) [Dempster et al., 1977]. For string s with m possible derivations, $d_1 - d_m$, we introduce indicator variables, $z_1 - z_m$, such that $z_i = 1$ if d_i is the derivation used when s was sampled. Otherwise, $z_i = 0$. The expected value of z_i can therefore be computed as follows:

$$\begin{aligned} E[z_i] &= 1 * p(z_i = 1) + 0 * p(z_i = 0) \\ &= p(z_i = 1) \\ &= p(d_i \text{ is the true derivation}) \\ &= \frac{p(d_i|G)}{\sum_j p(d_j|G)} \end{aligned}$$

In the expectation step, the indicator variables are used to compute expected counts:

$$\begin{aligned} \hat{c}(X \to \gamma|s) &= \sum_i E[z_i] c(X \to \gamma|d_i) \\ &= \frac{\sum_i p(d_i|G) c(X \to \gamma|d_i)}{\sum_j p(d_j|G)} \end{aligned} \quad (1)$$

In the maximization step, the expected counts are used to compute new maximum likelihood parameter estimates:

$$\hat{p}(X \to \gamma) = \frac{\hat{c}(X \to \gamma|E)}{\sum_\delta \hat{c}(X \to \delta|E)}$$

Iterating the E-step and the M-step is guaranteed to lead to a local maximum in the likelihood surface.

The only potential difficulty is that computing expected counts requires summing over all possible derivations of a string, of which there may be exponentially many. The Inside-Outside algorithm uses dynamic programming to compute these counts in polynomial time [Lari & Young, 1990]. Our discussion of the algorithm will follow the presentation in [Charniak, 1993].

For string s and non-terminal X, let $s_{i,j}$ denote the sub-string of s ranging from the i^{th} to the j^{th} position, and let $X_{i,j}$ denote the fact that non-terminal X roots the subtree that derives $s_{i,j}$. We can now define the inside probability, $\beta_X(i,j)$, as the probability that X will derive $s_{i,j}$. More formally:

$$\beta_X(i,j) = p(s_{i,j}|X_{i,j})$$

The outside probability, $\alpha_X(i,j)$, is the probability of deriving the string $s_{1,i-1} X s_{j+1,n}$ from the start symbol such that X spans $s_{i,j}$. More formally:

$$\alpha_X(i,j) = p(s_{1,i-1}, X_{i,j}, s_{j+1,n})$$

In the formula above, $n = |s|$. As figure 4 suggests, given that non-terminal X roots the sub-tree that derives $s_{i,j}$, the inside probability $\beta_X(i,j)$ is the probability of X deriving the part of s inside the sub-tree and the outside probability $\alpha_X(i,j)$ is the probability of the start symbol deriving the part of s outside the sub-tree.

How are α and β useful in parameter estimation? Rather than implementing equation 1 as a sum over derivations, we will soon see that knowing α and β makes it possible to compute expected counts by summing over all possible substrings of s that a given non-terminal can generate. For a string of length n there are $n(n-1)/2$ substrings, which is far fewer than the worst case exponential number of possible derivations.

For example, consider the somewhat simpler problem of computing the expected number of times X occurs in a derivation of string s. This non-terminal can potentially root sub-trees that generate any of the $n(n-1)/2$ substrings of s. The expected number of occurrences of X is thus given by the following sum:

$$\hat{c}(X) = \sum_{i,j} p(X_{i,j}|s)$$

This expression can be rewritten as follows in terms of inside and outside probabilities exclusively:

$$\begin{aligned}
\hat{c}(X) &= \sum_{i,j} p(X_{i,j}|s) \\
&= \frac{1}{p(s)} \sum_{i,j} p(X_{i,j}, s) \\
&= \frac{1}{p(s)} \sum_{i,j} p(s_{1,i-1}, s_{i,j}, s_{j+1,n}, X_{i,j}) \\
&= \frac{1}{p(s)} \sum_{i,j} p(s_{i,j}|X_{i,j}) p(s_{1,i-1}, X_{i,j}, s_{j+1,n}) \\
&= \frac{1}{p(s)} \sum_{i,j} \alpha_X(i,j)\beta_X(i,j)
\end{aligned}$$

The move from the first line to the second above is a simple application of the definition of conditional probability. We then expand s, apply the chain rule of probability, and finally substitute α and β.

Equation 1 requires $\hat{c}(X \to \gamma)$, not $\hat{c}(X)$. Suppose for the moment that our grammar is in Chomsky Normal Form. That is, all productions are of the form $X \to YZ$ or $X \to \sigma$ where X, Y, and Z are non-terminals and σ is a terminal. To compute $\hat{c}(X \to \gamma)$, rather than just summing over all possible substrings that X can generate, we sum over all possible substrings that X can generate and all possible ways that Y and Z can carve up the substring. Consider figure 4. If X generates $s_{i,j}$ and X expands to YZ, then concatenating the substring generated by Y with the substring generated by Z must yield $s_{i,j}$.

The expected counts for $X \to YZ$ are defined to be:

$$\hat{c}(X \to YZ) = \sum_{i,j,k} p(X_{i,j}, Y_{i,k}, Z_{k+1,j}|s)$$

It is easy to show that this is equivalent to:

$$\begin{aligned}
\hat{c}(X \to YZ) &= \frac{1}{p(s)} \sum_{i,j,k} \beta_Y(i,k)\beta_Z(k+1,j) \\
&\quad \alpha_X(i,j)p(X \to YZ) \qquad (2)
\end{aligned}$$

A complete derivation will be given in the next section when a formula for computing expected counts for graph grammars is presented.

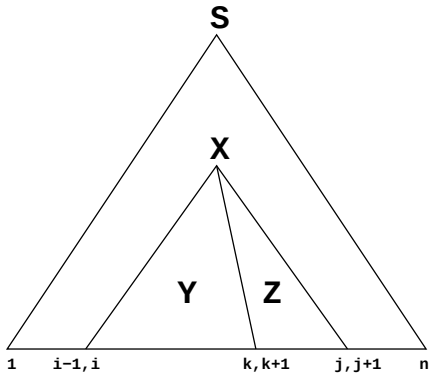


Figure 4: Given that X derives $s_{i,j}$ and that X expands to YZ, there are only $j-i+1$ ways that Y and Z can carve up $s_{i,j}$.

Clearly, evaluating equation 2 requires $O(n^3)$ computation, in addition to that required to compute α and β. For string grammars, tables of α and β values are computed via dynamic programming in $O(m^2 n)$ time where m is the number of non-terminals in the grammar. Section 4 will show a complete derivation of the formulas for computing α and β in the context of graph grammars.

4 Parameter Estimation for Graph Grammars

In this section we define and derive analogs of inside and outside probabilities for graph grammars. Just as α and β can be computed efficiently, top down and bottom up respectively, for string grammars by combining sub-strings, they can be computed efficiently for graph grammars by combining sub-graphs. While there are only polynomially many sub-strings of any given string, there in general can be exponentially many sub-graphs of any given graph. It turns out there there is a natural class of graphs [Lautemann, 1990] for which the number of sub-graphs that one must consider when computing α and β is polynomial in the size of the graph. For this type of grammar, maximum likelihood parameter estimates can be computed in polynomial time.

For non-terminal hyperedge X and graph g we define the inside probability $\beta_X(g)$ to be $p(g|X)$, the probability that X will derive g. Note that $\beta_S(g)$ is the probability of g in the distribution over graphs defined by the grammar. There are two cases to consider - either X derives g in one step, or X derives some other sub-graph in one step and g can be derived from that sub-graph in one or more steps:

$$\beta_X(g) = p(g|X)$$

$$= \quad p(X \rightarrow g) + \sum_{X \rightarrow \gamma} p(X \rightarrow \gamma) p(\gamma \stackrel{*}{\rightarrow} g) \quad (3)$$

In equation 3, we use $\rightarrow$ to denote derivation in one step via a production in the grammar and $\stackrel{*}{\rightarrow}$ to denote derivation in one or more steps.

The difficult part of evaluating equation 3 is computing $p(\gamma \stackrel{*}{\rightarrow} g)$. Because γ is the right-hand side of a production it can be an arbitrary hypergraph. Suppose γ has m hyperedges - $h_1, h_2, \ldots, h_m$. If γ can derive g, then there must be m graphs - $g_1, g_2, \ldots, g_m$ - such that h_i derives g_i for $1 \leq i \leq m$ and the graph that results from replacing each h_i with the corresponding g_i is isomorphic to g. Note that each g_i must be isomorphic to a sub-graph of g for this to occur. It is therefore theoretically possible to determine if γ (and thus X) can derive g by generating all possible sub-graphs of g, forming all ordered sets of these sub-graphs of size m, generating the graphs that result from substituting the sub-graphs in each ordered set for the hyperedges in γ, and testing to see if any of the resulting graphs are equal to g.

Because the sub-graphs are taken directly from g the equality test can be performed in polynomial time (i.e. a test for graph isomorphism is not required). However, there may be exponentially many sub-graphs. As stated earlier, we will restrict our attention to a robust class of graphs for which the number of sub-graphs one must consider is small (polynomial). Let's finish our derivation of $\beta_X(g)$ before getting to the details of computing it efficiently.

Let $\Psi(\gamma, g)$ be the set that results from computing all ordered sub-sets of size m of the set of all sub-graphs of g. Recall that γ is a hypergraph with m hyperedges (i.e. non-terminals). Let $\Psi_i(\gamma, g)$ be the i^{th} element of this set. Each element of Ψ represents a mapping of hyperedges in γ to structure in g. If any of these mappings yield g, then it is that case that γ can derive g.

To compute $p(\gamma \stackrel{*}{\rightarrow} g)$ we simply need to iterate over each element of $\Psi(\gamma, g)$ and compute the probability of the joint event that each of the h_i derive each of the g_i and sum this probability for each element that produces a graph equal to g. That is, for each $\Psi_i(\gamma, X)$ we need to compute:

$$p(h_1^i \stackrel{*}{\rightarrow} g_1^i, h_2^i \stackrel{*}{\rightarrow} g_2^i, \ldots, h_m^i \stackrel{*}{\rightarrow} g_m^i)$$

Because HR graph grammars are context free, derivations that start from different hyperedges are completely independent of one another [Courcelle, 1987]. Therefore, the probability of the joint event is equivalent to the product of the probabilities of the individual events. That is:

$$p(h_1^i \stackrel{*}{\rightarrow} g_1^i, h_2^i \stackrel{*}{\rightarrow} g_2^i, \ldots, h_m^i \stackrel{*}{\rightarrow} g_m^i) = \prod_{j=1}^{m} p(h_j^i \stackrel{*}{\rightarrow} g_j^i)$$

Combining the above with equation 3 yields an expression for $\beta_X(g)$ in terms of other inside probabilities:

$$\begin{aligned}
\beta_X(g) &= p(X \rightarrow g) + \sum_{X \rightarrow \gamma} p(X \rightarrow \gamma) p(\gamma \stackrel{*}{\rightarrow} g) \\
&= p(X \rightarrow g) + \sum_{X \rightarrow \gamma} p(X \rightarrow \gamma) \sum_i p(\Psi_i(\gamma, g)) \\
&= p(X \rightarrow g) + \sum_{X \rightarrow \gamma} p(X \rightarrow \gamma) \\
&\quad \sum_i p(h_1^i \stackrel{*}{\rightarrow} g_1^i, h_2^i \stackrel{*}{\rightarrow} g_2^i, \ldots, h_m^i \stackrel{*}{\rightarrow} g_m^i) \\
&= p(X \rightarrow g) + \sum_{X \rightarrow \gamma} p(X \rightarrow \gamma) \sum_i \prod_{j=1}^{m} p(h_j^i \stackrel{*}{\rightarrow} g_j^i) \\
&= p(X \rightarrow g) + \sum_{X \rightarrow \gamma} p(X \rightarrow \gamma) \sum_i \prod_{j=1}^{m} p(h_j^i | g_j^i) \\
&= p(X \rightarrow g) + \sum_{X \rightarrow \gamma} p(X \rightarrow \gamma) \sum_i \prod_{j=1}^{m} \beta_{h_j^i}(g_j^i) \quad (4)
\end{aligned}$$

Equation 5 makes it possible to compute inside probabilities in terms of other inside probabilities. Note that this computation can proceed bottom up because the sub-graphs considered in the inner sum, i.e. the recursive computation of $\beta_{h_j^i}(g_j^i)$, must be smaller than g because they are composed via γ to yield g. That is, one can compute β for sub-graphs containing one node, then sub-graphs containing two nodes, and so on. The number of levels in this bottom up computation is bounded by the size of g. The outer summation is linear in the number of productions in the grammar, and the product is linear in the maximum number of hyperedges in any right-hand side, which we assume to be bounded by a small constant. However, the inner sum iterates over all elements of Ψ, of which there can be exponentially many.

If the number of sub-graphs considered in the inner sum in equation 5 were polynomial, then all inside probabilities could be computed in polynomial time. Lautemann [Lautemann, 1990] defines a class of HR grammars for which this is the case, i.e. grammars with *logarithmic k-separability.*

The k-separability of graph g (see definition 3.2.3 in [Lautemann, 1990]) is the maximum number of connected components that can be produced by removing k or fewer nodes from g. This definition becomes useful for our current purposes when considered in conjunction with lemma 3.2.1 from [Lautemann, 1990]. To build intuition before stating the lemma, consider how you might try to determine if a hypergraph, γ, with a single hyperedge, h, can generate a given graph, g. Note that all of the nodes and edges in the hypergraph must appear in the final graph. You might therefore try all possible mappings of nodes and edges in the hypergraph to nodes and edges in the graph, and see if the hyperedge can generate the unmapped remainder of the graph.

The lemma says, essentially, that if replacing hyperedge h in hypergraph γ with graph g' yields graph g, then every connected component in g' minus the nodes in h is a connected component in g minus the nodes in h. That is, if you map the nodes in γ to nodes in g and then remove those mapped nodes from g and find the connected components in the resulting graph, you will

have enumerated (at least) all of the connected components in the sub-graph with which h should be replaced to derive g.

Therefore, rather than enumerating all possible sub-graphs of g to determine if $\gamma \stackrel{*}{\rightarrow} g$, we can form all possible mappings of nodes and edges in γ onto g, compute the connected components that result when the mapped nodes are removed from g, and consider only those sub-graphs that are combinations of these connected components. Because γ is the right-hand side of a production and we assume that its size is bounded by a small constant, the number of possible mappings of γ onto g is polynomial in the size of g. If we further assume that the k-separability of the graph is logarithmic, then the number of connected components formed for each mapping of γ onto g is $O(\log |g|)$ and there are only polynomially many possible combinations of connected components. In polynomial time we can compute all of the subgraphs that need to be considered in the inner sum of equation 5, of which there are polynomially many. All of the inside probabilities can therefore be computed in polynomial time.

Intuitively, bounded k-separability requires that graphs have bounded degree and be connected. Consider the language containing all star graphs, i.e. graphs containing n nodes where nodes $2-n$ have a single edge to node 1. If node 1 is removed, $n-1$ connected components are created. At the other extreme, consider a graph of n nodes and no edges. Removing any one node results in a graph with $n-1$ connected components. In both cases, the k-separability of the graph is linear in the size of the graph. For k-separability to have a lower bound, there must be a bound on node degree and the graph must be (mostly) connected.

We now turn to the derivation of the outside probability. Recall that the inside probability $\beta_X(g)$ is the probability that a non-terminal hyperedge labeled X will generate graph g. In practice, given a graph G, β values are computed for sub-graphs of G. That is, $\beta_X(g)$ is computed for values of g corresponding to different sub-graphs of some fixed graph G. The outside probability $\alpha_X(g)$ is the probability that the start symbol will generate the graph formed by replacing sub-graph g in graph G with a non-terminal hyperedge labeled X. It is called the outside probability because α is the probability of generating the graph structure in G outside the sub-graph generated by X. Note that the quantity $\alpha_X(g)\beta_X(g)$ is the probability of generating G in such a way that nonterminal X generates sub-graph g.

How might non-terminal X become responsible for generating sub-graph g? Suppose Y is a non-terminal, $Y \rightarrow \gamma$ is a production in the grammar, and γ contains a hyperedge labeled X. Further, let g' be a subgraph of G that contains g. If Y is responsible for generating g', then it could be the case that X generates g and the remainder of γ generates the remainder of g'. That is, we can compute outside probabilities from outside probabilities of larger subgraphs.

The above is formalized in equation 6. The outer sum ($\sum_{Y \rightarrow \gamma}$) is over all productions in the grammar. The next sum ($\sum_{i=1}^{m}$) is over all hyperedges in γ, of which there are m. The first term in that sum ($\delta(h_i = X)$) takes on the value 1 when the i^{th} hyperedge in γ is labeled X and takes on the value 0 otherwise. Collectively, the first two sums and the first term iterate over all hyperedges labeled X in all right-hand sides.

The next sum ($\sum_j$) indexes into $\Psi(\gamma, G)$, the set of all ordered subsets of size m of subgraphs of G. The first term in that sum ($\delta(g_i^j = g)$) selects those elements of $\Psi(\gamma, G)$ that have g in a position that matches it up with an X in γ. The next term multiples by the probability that Y actually generated the subgraph of G represented by the union of the elements of $\Psi_j(\gamma, G)$.

The next term ($p(Y \rightarrow \gamma)$) is the probability that Y, which generates $\Psi_j(\gamma, G)$, expands to γ, whose i^{th} hyperedge is an X and is mapped to g in $\Psi_j(\gamma, G)$. The final product is over all hyperedges in γ except the i^{th} hyperedge, where each term is the probability that the hyperedge will generate the subgraph to which it is mapped via $\Psi_j(\gamma, G)$.

$$\alpha_X(g) = \sum_{Y \rightarrow \gamma} \sum_{i=1}^{m} \delta(h_i = X) \sum_j \delta(g_i^j = g) \alpha_Y(\Psi_j(\gamma, G)) \; p(Y \rightarrow \gamma) \prod_{k \neq i} \beta_{h_k}(g_k^j) \quad (5)$$

Equation 6 formalizes the notion that X can be responsible for g only if it is generated by expanding a hyperedge that is responsible for a sub-graph containing g. Inside probabilities can be computed from the top down, with the base case being $\alpha_S(G) = 1$. That is, with probability 1 the start symbol is responsible for generating any graph in the language of the grammar.

As with β, all of the sums and products are polynomial in the size of the graph except the one that iterations over the elements of $\Psi(\gamma, G)$. However, as with β, if the graph has logarithmic k-separability, there are only polynomially many elements of $\Psi(\gamma, G)$ to consider. Therefore, all outside probabilities can be computed in polynomial time.

Finally, using the inside and outside probabilities to estimate the parameters of the grammar is a relatively straightforward modification to the iteration used by IO for string grammars. Due to lack of space, further details will not be provided here and the interested reader is referred to [Charniak, 1993] for more information on the nature of that computation.

5 Preliminary Empirical Results

This section reports the results of some simple, preliminary experiments with an implementation of PEGG. Let $G = (\mathcal{S}, \theta)$ be a stochastic context-free HR graph grammar with structure $\mathcal{S}$ and parameters θ. Let E be a set of training examples created by sampling from the probability distribution over graphs defined by G. Given $\mathcal{S}$

and E, the goal of PEGG is to obtain a set of parameters, $\hat{\theta}$, such that $p(E|S,\hat{\theta})$ is maximized. We used the grammar shown in figure 1 for $\mathcal{S}$, and the true parameters were $\theta = (0.6, 0.2, 0.2)$. That is, the probability of expanding a hyperedge labeled S with the first production is 0.6, with the second production is 0.2, and with the third production is 0.2.

In the first experiment we sampled 1, 5, and 10 graphs from the grammar and ran PEGG on these sample. The learned parameters are shown in table 1. In all cases the parameters appear to be "reasonable", but they do deviate from the desired parameters. This might be due to the fact that the samples are small and are therefore not representative of the true distribution over graphs defined by the grammar. To test this hypothesis, we took another sample of 10 graphs for which the estimated parameters deviated significantly from the true parameters. Given the derivations of the 10 graphs, it was a simple matter to count the number of times the various productions were applied and manually compute maximum likelihood parameters. Both the estimated parameters and the ML parameters are shown in table 2. Note that the sample of 10 graphs was clearly not representative of the distribution over graphs defined by the true parameters. The Kullback-Liebler divergence between $\theta = (0.6, 0.2, 0.2)$ and $\theta_{ML} = (0.75, 0.15, 0.10)$ is 0.8985. However, PEGG did a good job of estimation. The KL divergence between θ_{PEGG} and θ_{ML} is 0.007, two orders of magnitude less than the divergence with the true parameters.

Table 1: Parameters estimated by PEGG for samples of size 1, 5, and 10.

$\|E\|$	θ_1	θ_2	θ_3
1	0.5714	0.2857	0.1429
5	0.6206	0.2168	0.1626
10	0.6486	0.2973	0.0541

Table 2: Estimated parameters and manually computed ML parameters for a sample of size 10.

	PEGG	ML
θ_1	0.7368	0.75
θ_2	0.1579	0.15
θ_3	0.1053	0.10

Finally, to determine if PEGG was finding parameters that actually maximize the likelihood of the data, we computed the log-likelihood of a sample of 5 graphs given the true parameters and the parameters estimated for that sample. The log-likelihood of the data given the true parameters was -9.38092, and it was -9.40647 given the estimated parameters, a difference of less than three-tenths of one percent.

6 Conclusion

This paper introduced the Parameter Estimation for Graph Grammars (PEGG) algorithm, the first algorithm for estimating the parameters of stochastic context-free hyperedge replacement graph grammars. PEGG computes inside and outside probabilities in polynomial time for graphs with logarithmic k-separability. In addition, PEGG uses these probabilities to compute maximum likelihood parameters for a fixed grammar structure and a sample of graphs drawn from some probability distribution over graphs.

Despite that fact that graph grammars have been an active area of research since the late 1960's, almost no work has dealt with stochastic graph grammars. One notable exception is [Mosbah, 1994], which explores the properties of graphs sampled from stochastic graph grammars.

There are only a handful of papers that directly address the problem of learning graph grammars, and none other than the current paper that leverage the vast body of work on inferring string grammars from data. [Bartsch-Sprol, 1983] describes an enumerative (i.e. computationally infeasible) method for inferring a restricted class of context-sensitive graph grammars. [Jeltsch & Kreowski, 1991] describes an algorithm for extracting common hyperedge replacement sub-structures from a set of graphs via merging techniques. This work is similar to that reported in [Jonyer et al., 2002] in which merging techniques were used to extract node replacement sub-structures from a set of graphs. Fletcher [Fletcher, 2001] developed a connectionist method for learning regular graph grammars. To the best of our knowledge, our paper is the first to present a formally sound algorithm for computing maximum likelihood parameter estimates for a large class of HR graph grammars.

Future work will involve developing an approach to inferring the structure of HR graph grammars based on Bayesian model merging techniques, similar to those we developed for node replacement grammars [Doshi et al., 2002]. In combination with the PEGG algorithm described in this paper the result will be a powerful tool for inferring HR graph grammars from data. In addition, we are considering applications of this tool in the domain of bioinformatics, such as refining initial protein clusters based on primary structure (linear sequences of nucleotide) by learning graph grammars based on secondary structure (the arrangement of alpha helices and beta sheets in three-dimensional space).

References

Bartsch-Sprol, B. (1983). Grammatical inference of graph grammars for syntactic pattern recognition. In H. Ehrig, M. Nagl and G. Rozenberg (Eds.), *Proceedings of the second international workshop on graph grammars and their applications to computer science.* Springer Verlag.

Charniak, E. (1993). *Statistical language learning.* MIT Press.

Cook, D., Holder, L. B., Su, S., Maglothin, R., & Joyner, I. (1994). Structural mining of molecular biology data. *IEEE Engineering in Medicine and Biology, 20*, 231–255.

Courcelle, B. (1987). An axiomatic definition of context-free rewriting and its application to NLC graph grammars. *Theoretical Computer Science, 55*, 141–181.

Courcelle, B. (1997). The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg (Ed.), *Handbook of graph grammars and computing by graph transformation: Foundations*. World Scientific Publishing Company.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B, 39*, 1–38.

Doshi, S., Huang, F., & Oates, T. (2002). Inferring the structure of graph grammars from data. . Proceedings of the International Conference on Knowledge-Based Computer Systems.

Drewes, F., Kreowski, H. J., & Habel, A. (1997). Hyperedge replacement graph grammars. In G. Rozenberg (Ed.), *Handbook of graph grammars and computing by graph transformation: Foundations*. World Scientific Publishing Company.

Ehrig, H., Engels, G., Kreowski, H.-J., & Rozenberg, G. (Eds.). (1999). *Handbook of graph grammars and computing by graph transformation: Applications, languages and tools*. World Scientific Publishing Company.

Engelfriet, J., & Rozenberg, G. (1997). Node replacement graph grammars. In G. Rozenberg (Ed.), *Handbook of graph grammars and computing by graph transformation: Foundations*. World Scientific Publishing Company.

Fletcher, P. (2001). Connectionist learning of regular graph grammars. *Connection Science, 13*, 127–188.

Hong, P., & Huang, T. S. (2002). Spatial pattern discovery by learning a probabilistic parametric model from multiple attributed relational graphs. *Journal of Discrete Applied Mathematics*.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, languages and computation*. Addison-Wesley Publishing Company.

Immerman, N. (1999). *Descriptive complexity*. Springer.

Jeltsch, E., & Kreowski, H. J. (1991). Grammatical inference based on hyperedge replacement. *Lecture Notes in Computer Science, 532*, 461–474.

Jonyer, I., Holder, L. B., & Cook, D. J. (2002). Concept formation using graph grammars. *Working Notes of the KDD Workshop on Multi-Relational Data Mining*.

Lari, K., & Young, S. J. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language, 4*, 35–56.

Lautemann, C. (1990). The complexity of graph languages generated by hyperedge replacement. *Acta Informatica, 27*, 399–421.

Mosbah, M. (1994). Properties of random graphs generated by probabilistic graph grammars. *Proceedings of the The Fifth International Workshop on Graph Grammars and their Application to Computer Science*.

Office of Technology Assessment, U. C. (1995). Information technologies for control of money laundering. OTA-ITC-360.

Pattison, P. E. (1993). *Algebraic models for social networks*. Cambridge University Press.

Stolcke, A. (1994). *Bayesian learning of probabilistic language models*. Doctoral dissertation, University of California, Berkeley.

Aggregation and Concept Complexity in Relational Learning

Claudia Perlich and Foster Provost
NYU Stern School of Business
44 West 4th Street
New York, NY 10012, U.S.A.
cperlich@stern.nyu.edu, fprovost@stern.nyu.edu

Traditional, flat-table learning algorithms have been scrutinized for many years from many angles, and are well understood with respect to their applicability and expressive power. Much less can be said about relational learning approaches. One main reason is the lack of clarity of the space of possible relational concepts.

We believe that a workshop that brings together people with a variety of perspectives on relational learning represents a unique opportunity to develop a formal characterization of relational concepts. Such formalization would contribute to research on relational learning in a number of ways.

1. It would provide a framework for the theoretical analysis and comparison of relational learners with respect to their ability to express and learn certain concepts.
2. It could thereby provide guidance in the choice of a method for a particular domain.
3. It would be a valuable tool for addressing the issue of model and representation complexity.
4. It would help to clarify the problem domain that a practitioner/research effort is targeting.
5. It would help to specify the scope of generalization of scientific claims.

We argue for a hierarchy of increasingly more general relational concept classes, conveniently placing standard, single-table ("propositional") learning algorithms as learning concepts in the most specialized class. Some other, more general, cases of relational concepts have been examined already, for example attribute hierarchies (Almuallim et al., 1995). The most general (and complex) class would capture global concepts that include the entire relational structure and all object attributes.

We expect learners developed for more specialized concept classes to apply broadly within the class, but of course to be suboptimal (biased) when a more general class is required. Nevertheless, it may be that the biased learning method actually performs better than an unbiased counterpart, for some problems. This is in direct analogy to what has been observed time and time again in machine learning. For example, linear models (including naïve Bayes) learn a more restricted concept class than does tree induction. However, they are extremely useful. In some situations, linear models are preferable to tree induction even for problems where the true, underlying model is represented better by a tree (Domingos & Pazzani, 1996; Perlich, Simonoff, & Provost, 2003). Relational learning systems (e.g., ILP systems) often perform suboptimally on purely propositional tasks, even when in principle they are capable of representing the true concept.

Furthermore, given the extreme complexity of higher-level relational concept classes, it is likely that modeling approaches designed for lower levels will have broader expressive power within the lower-level class. For example, ILP systems can represent very complex relational concepts. However, they cannot take full advantage of statistical properties like relational autocorrelation (Jensen and Neville 2000), which can be extremely useful for modeling in relational domains.

A well-designed concept-class hierarchy will also facilitate a "bi-directional search" approach to relational learning research. Although we are not aware of it being framed as such, this type of approach already is being taken. Some researchers are asking how ILP systems can be extended to deal more robustly with more specialized concept classes, and recently there has been a surge of interest in generalizing propositional algorithms (e.g., Bayesian networks, decision trees, logistic regression, and naïve Bayes). A concept-class hierarchy will clarify and to some degree quantify the nature of extensions of propositional learning methods. For example, these extensions of propositional methods typically cannot learn concepts as general as can ILP systems. However, they may be more robust at learning concepts at the lower level. Research results along these lines are much more satisfying than saying "my relational Foo algorithm is better than FOIL on the domains I've tested."

An Aggregation-based framework

Moving upward from propositional algorithms to algorithms that can learn more complex relational concepts, we must address two main issues: (i) how to explore related objects in secondary tables and (ii) how to aggregate the bags of related objects that are found. We conjecture that the formalization of relational concepts must (at least) reflect these two main issues:

- The definition of "relatedness"

- The aggregation of bags of related objects

We have proposed (Perlich and Provost 2003) a first formalization of a relational concept hierarchy that is increasing in the complexity of aggregation with respect to: the number of related objects, the number of object attributes, dependencies between object attributes, number of different object types, and the locality of the concept.

Results using the framework

The paper also reports an empirical study that compares experimentally several different aggregation approaches and assesses the generalization ability with respect to the maximum complexity that the approach could express. In particular we compared (in order of increasing complexity) simple propositional methods without background knowledge, the somewhat more complex case of an abstraction hierarchy without aggregation, simple single-attribute aggregation (most common categorical value, counts, proportions), complex single-attribute aggregation using target-dependent set distances, and logic-based dependent attribute aggregation. The relative performances suggest an optimal aggregation complexity level (target-dependent attribute aggregation) above which the performance decreases.

We conclude from our results so far that the expressive power (and performance) of a relational learning algorithm is strongly related to the aggregation methods applied. Increasing the complexity of the relational aggregation shows significant improvements in the generalization performance. In fact, the best method we tested is a transformation-based approach that uses aggregation to construct a single-table learning task from relational data, and then takes advantage of the sophistication of traditional propositional learners.

Further Implications and Work

Having identified aggregation as a major driver of generalization performance suggests that more efforts should be made to dissect complex relational learning systems into their components and to identify the source of generalization power.

As a research field we should work our way up stepwise through increasing complexity, drawing from the knowledge and experience of lower levels, rather than jumping to very expressive model classes that (currently) suffer from massive search problems.

Additional outstanding tasks are the

- improvement of the hierarchy of relational concepts,
- exploration of "relatedness" and alternative methods of selecting related objects,
- identification of other components of concepts and of algorithms (besides exploration and aggregation),
- undertaking of more comparative studies and acquiring additional benchmark datasets.

Acknowledgements

We are grateful to Jeff Simonoff and Sofus Macskassy for valuable comments and discussions.

This work is sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-01-2-585.

References

Almuallin, H; Y. Akiba, and S. Kaneda (1995). On handling tree-structured attributes in decision tree learning. *In Proceedings of the Twelfth International Conference on Machine Learning 1995*. 12-20. Morgan Kaufmann.

Domingos, P. and P. Michael (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning 29(2-3), 103-130*

Jensen, D. and J. Neville (2002). Autocorrelation and Linkage Cause Bias in Evaluation of Relational Learners. *In Proceedings of The Twelfth International Conference on Inductive Logic Programming (ILP 2002).* Springer-Verlag

Perlich, C. and F. Provost (2003). Aggregation-based Feature Invention and Relational Concept Classes. *In Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining(KDD-2003)*

Perlich, C., F. Provost, and J. Simonoff (2003). Tree Induction vs. Logistic Regression: A Learning-curve Analysis. *To appear in the Journal of Machine Learning Research*. Preliminary version: *CeDER Working Paper #IS-01-02,* Stern School of Business, New York University

Statistical Relational Learning for Link Prediction

Alexandrin Popescul and **Lyle H. Ungar**
Department of Computer and Information Sciences
University of Pennsylvania
Philadelphia, PA 19104
{popescul,ungar}@cis.upenn.edu

Abstract

Link prediction is a complex, inherently relational, task. Be it in the domain of scientific citations, social networks or hypertext links, the underlying data are extremely noisy and the characteristics useful for prediction are not readily available in a "flat" file format, but rather involve complex relationships among objects. In this paper, we propose the application of our methodology for *Statistical Relational Learning* to building link prediction models. We propose an integrated approach to building regression models from data stored in relational databases in which potential predictors are generated by structured search of the space of queries to the database, and then tested for inclusion in a logistic regression. We present experimental results for the task of predicting citations made in scientific literature using relational data taken from CiteSeer. This data includes the citation graph, authorship and publication venues of papers, as well as their word content.

1 Introduction

Link prediction is an important problem arising in many domains. Web pages, computers, scientific publications, organizations, people and biological molecules are interconnected and interact in one way or another. Being able to predict the presence of links or connections between entities in a domain is both important and difficult to do well. We emphasize two important characteristics of such domains: i) their nature is inherently multi-relational, making the standard "flat" file domain representation inadequate, and ii) such data is often very noisy or partially observed. For example, in the domain of scientific publications, documents are cited based on many criteria, including their topic, conference or journal, and authorship, as well as the already existing citation structure. All attributes contribute, some in fairly complex ways.

The characteristics of the task suggest statistical learning for building robust models from noisy data and relational databases as a natural way to represent and store such data. Difficulties arise from the fact that the standard statistical learning algorithms assume one-table "flat" domain representation. Such algorithms are presented with a set of possible predictors, and a model selection process makes decisions regarding their inclusion into a model. Thus, the process of feature creation is decoupled from feature selection, and is often performed manually. Moreover, it is not always obvious what features should be generated. Thus, it is crucial to provide statistical modeling techniques with an integrated functionality to navigate richer data structures in order to discover potentially new and complex sources of relevant evidence, features not readily available in single tables and not always immediately obvious to humans.

One approach that might be considered is to generate the full join of a database for a one-table learning method. This is both impractical and incorrect—the size of the resulting table is prohibitive, and the notion of an object corresponding to a training example is lost, being represented by multiple rows. Moreover, the entries in the full join table will be *atomic* attribute values, rather than values resulting from arbitrarily complex queries, what we desire for our features. Preserving the relational representation also allows the introduction of intelligent search heuristics that explore only subspaces of the possible search space.

Our method of statistical relational learning integrates standard statistical modeling, here logistic regression, with a process for systematically generating features from relational data. We formulate the feature generation process as *search in the space of relational database queries*. The richness of this space or *space bias* potentially can be chosen at the desired level of complexity by specifying the types of queries allowed in the search. Aggregation or statistical operations, groupings, richer join conditions, or *argmax*-based queries can all be considered as part of search. Thus, the search space allows the discovery of complex and interesting relationships.

In this paper, we apply our method of statistical relational learning [Popescul *et al.*, 2003] to the problem of citation prediction in the domain of scientific publications. Link prediction models in this domain can be used as a citation recommender service. This service can potentially be deployed to recommend citations to users who provide the abstract, names of the authors and possibly a partial reference list of a paper in progress. In addition to prediction, the learned features have an explanatory power, providing insights into the nature of the citation graph structure. We use the data from CiteSeer

(a.k.a. ResearchIndex),[1] an online digital library of computer science papers [Lawrence *et al.*, 1999]. CiteSeer contains a rich set of relational data, including the text of titles, abstracts and documents, citation information, author names and affiliations, conference or journal names.[2]

2 Methodology

Our method of statistical relational learning couples two main processes: generation of feature candidates from relational data and their selection with statistical model selection criteria. Relational feature generation is a search problem. It requires formulation of the search in the space of queries to a relational database. We introduce notation first.

Throughout this section we use the following fixed schema:[3]

Citation(*from*:Document, *to*:Document),
Author(*doc*:Document, *auth*:Person),
PublishedIn(*doc*:Document, *vn*:Venue),
WordCount(*doc*:Document, *word*:Word, *cnt*:Int).

We use extended relational algebra notation to denote aggregations. Aggregation functions are subscripted with the corresponding attribute name if applied to an individual column, or are used without subscripts if applied to entire tables. For example, an average count of the word "learning" in documents cited from a learning example document d, a potentially useful type of feature in document classification, is denoted as:

$$ave_{cnt}[\sigma_{word='learning' \wedge from='d'} (Citation \bowtie_{to=doc} WordCount)]$$

The duplicates in the column *cnt* are not eliminated, unless an explicit projection of that column is performed before aggregation takes place. When ambiguous, column names are resolved with relation names. We abbreviate relation names with their first letter, and in the cases of joins involving more than one instance of the same relation, the relation name is suffixed with a numeral. For example, the number of common documents that both documents d^1 and d^2 cite is:

$$count[\sigma_{C1.from='d^1' \wedge C2.from='d^2'} (C1 \bowtie_{C1.to=C2.to} C2)].$$

This feature is an example of a feature useful in link prediction. It asks a question about a target pair of documents $< d^1, d^2 >$. When learning n-ary targets, we superscript d with a corresponding attribute index. Queries may be about just one of the documents in a target pair. For example:

$$count[\sigma_{to='d^2'}(C)]$$

is the number of times document d^2 is cited. Larger values of this feature increase the prior probability of d^1 citing d^2, regardless of what d^1 is.

[1] http://citeseer.org/

[2] Publication venues are extracted by matching information with the DBLP database: http://dblp.uni-trier.de/

[3] Domains, or types, used here are different from the basic SQL types. In implementation, these domains are specified in addition to the basic SQL types to guide the search process more efficiently.

2.1 Relational Feature Generation

We generate features by searching the space of relational database queries. The main principle of our search formulation is based on the concept of *refinement graphs* [Shapiro, 1983] which are widely used to search the space of first-order logic clauses. The search of refinement graphs starts with most general clauses and progresses by refining them into more specialized ones. Refinement graphs are directed acyclic graphs specifying the search space of the first-order logic queries. The space is constrained by specifying legal clauses (e.g. disallowing recursion and negation), and then structured by partial ordering of clauses, using a syntactic notion of generality (θ-subsumption [Plotkin, 1969]). A search node is expanded, or refined, applying *a refinement operator* to produce its most general specializations. Inductive logic programming systems using refinement graph search, usually apply two refinement operators: i) adding a predicate to the body of a clause, involving one or more variables already present, and possibly introducing one or more new variables, or ii) a single variable substitution, [Dzeroski and Lavrac, 2001]. In relational algebra these refinements correspond to adding an equijoin with a new relation instance, or performing a selection based on a condition of equality with a constant. In contrast to learning logic clauses, statistical learning is not limited to binary logic valued attributes.

Our first extension introduces aggregation or statistical operations into the search space. A query in relational algebra results in a table of all attribute values satisfying it, rather than a `true/false` value. Query results are aggregated to produce scalar numeric values to be used as features in statistical learning. Although there is no limit to the number of aggregation functions one may try, e.g. square root of the sum of column values, logarithm of their product etc., we expect a few of them being most useful, such as `count`, `ave`, `max`, `min`, `mode`, and `empty`. Aggregations can be applied to a whole table or to individual columns, as appropriate given type restrictions, e.g. `ave` cannot be applied to a column of a categorical type. Adding aggregation operators results in a much richer search space. Binary logic-based features are also included through the `empty` aggregation operation. The situations when aggregation operations are not defined, e.g. the average of an empty set, are resolved by introducing an interaction term with a 1/0 not-defined/defined feature.

The results of aggregation operations may be factored into further search. For example, we may want to ask how many co-authors the most cited author in a given conference c has (including him/herself). The following aggregation:

$$mostCitedAuth = mode_{auth}[(\sigma_{vn='c'}((P \bowtie_{P.doc=from} C) \bowtie_{to=A.doc} A))]$$

is used in:

$$count[\pi_{A2.auth}(\sigma_{A1.auth='mostCitedAuth'} (A1 \bowtie_{A1.doc=A2.doc} A2))].$$

Richer selection or join conditions, not necessarily conjuncts of equality conditions, can also be made part of the search space. The search space is potentially infinite, but not all subspaces will be equally useful. We propose the use of

sampling from subspaces of the same type performed at the time of node expansions to decide if more thorough exploration of that subspace is promising, or if the search should be more quickly refocused on other subspaces.

Our current implementation considers the search space covering queries with equijoins, equality selections and aggregation operations. Aggregates are considered for model inclusion at each node, but are not factored into the further search. Figure 1 presents a fragment of the search space using relations `Author`, `Citation` and `PublishedIn` for the link prediction task.

Logistic regression [Hosmer and Lemeshow, 1989] is used for binary classification problems. Model parameters/regression coefficients are learned to maximize the likelihood function, i.e. the probability that the training data is generated by a model with these coefficients. More complex models will results in higher likelihood values, but at some point will likely overfit the data, resulting in poor generalization. A number of criteria aiming at striking the balance between optimizing the likelihood of training data and model complexity have been proposed. Among the more widely used is the Bayesian Information Criterion (BIC) [Schwartz, 1979], which works by penalizing the likelihood by a term that depends on model complexity. We use stepwise model selection to find a model which generalizes well by adding one predictor at a time as long as the BIC can still be improved.

3 Tasks and Data

Learning from relational data for link prediction differs in several important aspects from other learning settings. Relational learning, in general, requires a quite different paradigm from "flat" file learning. The assumption that the examples are independent is violated in the presence of relational structure; this can be addressed explicitly [Jensen and Neville, 2002; Hoff, 2003], or implicitly, as we do here, by generating more complex features which capture relational dependencies. When the right features are used, the observations are conditionally independent given the features, eliminating the independence violation.

In our link prediction setting, a learning example class label indicating the presence or absence of a link between two documents is information of the same type as the rest of the link structure which can solely be used for prediction. In some sense, it may be instructive to view this setting as *hybrid model and memory based* learning. We build a formal statistical model, but prediction of future data points requires database access, as each selected feature is a database query. Thus, an important aspect, more so than in attribute-value learning, is what information about new examples will be available at the time of prediction and how missing or changing background information would affect the results.

Consider the following two scenarios for prediction of links between objects in a domain:

- The identity of all objects is known. Only *some* of the link structure is known. The goal is to predict unobserved links, from existing link structure alone or also using information about other available object attributes.
- New objects arrive and we want to predict their links to other existing objects. What do we know about new objects? Perhaps, we know some of their links, and want to predict the other. Alternatively, we might not know any of the links, but know some other attributes of the new objects.

In the latter case, when none of the partial link structure of the new objects is known, and prediction is based solely on other attributes, e.g. only authorship and word content, feature generation would have to be controlled to not produce features based on immediate links, but use them when referring to the links in already existing background knowledge.

In this paper, we perform experiments for the first scenario. The data for our experiments was taken from CiteSeer [Lawrence *et al.*, 1999]. CiteSeer catalogs scientific publications available in full-text on the web in PostScript and PDF formats. It extracts and matches citations to produce a browsable citation graph. The data we used contains 271,343 documents and 1,092,200 citations.[4] Additional information includes authorship and publication relations.[5] We use the following schema:

Citation(*from*:`Document`, *to*:`Document`),
Author(*doc*:`Document`, *auth*:`Person`),
PublishedIn(*doc*:`Document`, *vn*:`Venue`).

The training and test sets are formed by sampling citations (or absent citations for negative examples) from the citation graph. We perform learning on five datasets. Four of the datasets include links among all documents containing a certain query phrase, and the fifth data set covers the entire collection. Note that the background knowledge in the first four datasets also includes all other links in the full collection; only training and test links are sampled from the subgraph induced by document subsets. Table 1 contains the summary of the datasets.

The detailed learning setting is as follows:

- Populate three relations `Citation`, `Author` and `PublishedIn` initially with *all* data.
- Create training and test sets of 5,000 examples each by i) randomly sampling 2,500 citations for training and 2,500 citations for testing from those in column `# Links` of the Table 1; and ii) creating negative examples by sampling from the same subgraph also 2,500/2,500 train/test of "empty" citations, i.e. pairs of documents not citing each other.
- *Remove* test set citations from the `Citation` relation; but not the other information about the documents involved in the test set citations. For example, other citations of those documents are not removed.

[4] This data is part of CiteSeer as of August 2002. Documents considered are only non-singleton documents out of the total of 387,703. Singletons are documents which both citation indegree and outdegree registered in CiteSeer are zero.

[5] The authorship information is known for 218,313 papers, and includes 58,342 authors. Publication venues are known for 60,646 documents. The set of venues consists of 1,560 conferences and journals.

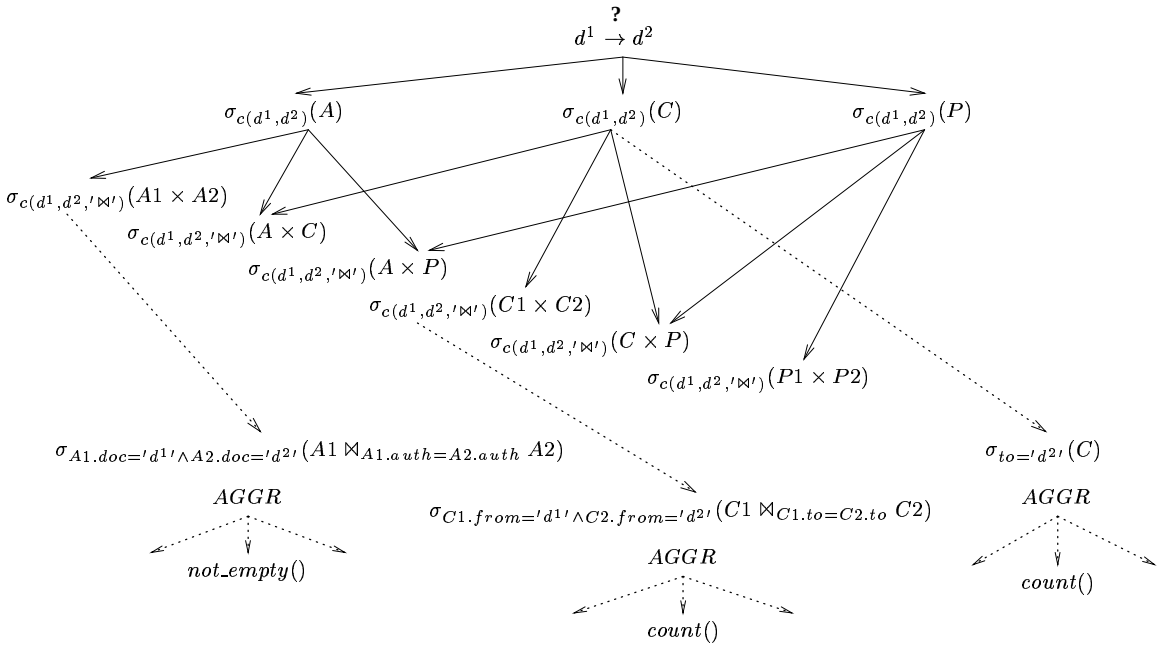


Figure 1: Fragment of the search space and examples. The select condition c is a boolean function specifying join conditions and equality conditions for referencing learning target-pair $< d^1, d^2 >$. Each node is a database query about d^1, d^2 or both (relation names are first letter abbreviations of `Citation`, `Author` and `PublishedIn` and the number if more that one instantiation of the same relation is involved. $AGGR$ denotes features resulting in aggregation operations at the corresponding search graph node).

Table 1: Dataset summaries.

Dataset	# Docs	# Links	Density (10^{-2}%)
"artificial intelligence"	11,144	16,654	1.3
"data mining"	3,424	6,790	5.8
"information retrieval"	5,156	8,858	3.3
"machine learning"	6,009	11,531	3.2
entire collection	271,343	1,092,200	0.1

Table 2: Training and test set accuracies (%). 5,000 train/test examples; balanced priors.

Dataset	BK: Citation		BK: All	
	Train	Test	Train	Test
"artificial intelligence"	90.24	89.68	92.60	92.14
"data mining"	87.40	87.20	89.70	89.18
"information retrieval"	85.98	85.34	88.88	88.82
"machine learning"	89.40	89.14	91.42	91.14
entire collection	92.80	92.28	93.66	93.22

- *Remove* training set citations from the `Citation` relation, so as not to include the actual answer in the background knowledge.
- Learning is performed i) using `Citation` relation only, or ii) using all three relations `Citation`, `Author` and `PublishedIn`.

The positive and negative classes in this task are extremely unbalanced. We ignore the lack of balance at the training phase; at the testing phase we perform additional *precision-recall* curve analysis for larger negative class priors. The next section reports experimental results.

4 Results

We start by presenting the results for the balanced class priors test scenario, and continue with the analysis of the unbalanced class settings. Two sets of models are learned for each dataset: i) using only `Citation` background knowledge, and ii) using all three relations `Citation`, `Author` and `PublishedIn`.

When only `Citation` background knowledge is used the average test set accuracy in five datasets is 88.73% and when all relations are used the average increases to 90.90%.[6] In both cases the search explored features involving joins of up to three relations. It is not unreasonable to expect that even better models can be built if we allow the search to progress further. Table 2 details the performance in each dataset. The largest accuracy of 93.22% is achieved for the entire CiteSeer dataset. Even though this is the largest and the most sparse dataset, this is not surprising because, since the features are not domain specific and rely on the surrounding citation structure, this dataset retains more useful "supporting link structure" after some of them are removed to serve as training and testing examples (Section 3).

In the experiments using only the `Citation` relation the average number of features selected is 32; 13 of the selected

[6] Using the predicted probability of 0.5 as the decision cut-off in logistic regression.

features are the same across all five datasets. When all three relations `Citation`, `Author` and `PublishedIn` are used the average number of selected features is 40, with 14 features common to all five datasets. In addition to more obvious features, such as d^1 is more likely to cite d^2 if d^2 is frequently cited, or if the same person co-authored both documents, or if d^1 and d^2 are co-cited, or cite the same papers[7], we learned some more interesting features. For example, a document is more likely to be cited if it is cited by frequently cited documents. Locally, this effectively learns the concept of an authoritative document [Page *et al.*, 1998; Kleinberg, 1999]. Or, the following feature, selected in all models:

$$count[\pi_{C2.to}(\sigma_{C1.to='d^2'}(C1 \bowtie_{C1.from=C2.from} C2))]$$

increases the probability of a citation if d^2 is co-cited with many documents. Since this feature is selected in addition to the simple citation count feature, it could mean that either d^2 appears more often in reviews, which tend to have longer lists of references, or it is cited from documents having smaller overlap among their references, which is more probable if they belong to different communities.

We compare the above results to the models trained on only binary features, i.e. when using only the `empty` aggregation function on the entire table. Such features are the logic-based features from the original formulation of refinement graph search; or, in other words, propositionalized inductive logic programming features with logistic regression feature selection [Popescul *et al.*, 2002]. The binary features resulted in models with lower out-of-sample accuracies in all datasets. On average the accuracy with only binary features is 2.52 percentage points lower in models using `Citation` relation, and 2.20 percentage points lower in models using all three relations. The decrease of accuracy is significant at the 99% confidence level in both cases according to the t-test.

The class priors are extremely unbalanced, due to the sparsity of the citation structure. The citation graph of the "artificial intelligence" dataset, for example is only 1.34×10^{-4} dense; that means that for one citation between two documents there are more than 7,000 non-existing citations, thus there are more than 7,000 times as many negative examples as there are positive. We perform the precision-recall curve analysis of our models trained with balanced class priors for testing situations with increased negative class proportions.

We vary the ratio k of the number of negative to the number positive examples used at testing. The ratio of one corresponds to the initial balance. We use for illustration the "artificial intelligence" dataset and the model trained using all three relations. New larger sets of negative examples are sampled with replacement from all "non-existing" links between documents in this dataset. Figure 2 presents precision-recall curves for k = 1, 10 and 100. As k increases the precision falls for the same levels of recall. Reducing the negative class prior should be performed when possible by filtering out obviously negative examples, for example by using a text based similarity, or other measures appropriate for a given task.

[7] These two features correspond to the concepts of co-citation and bibliographic coupling used in bibliometrics.

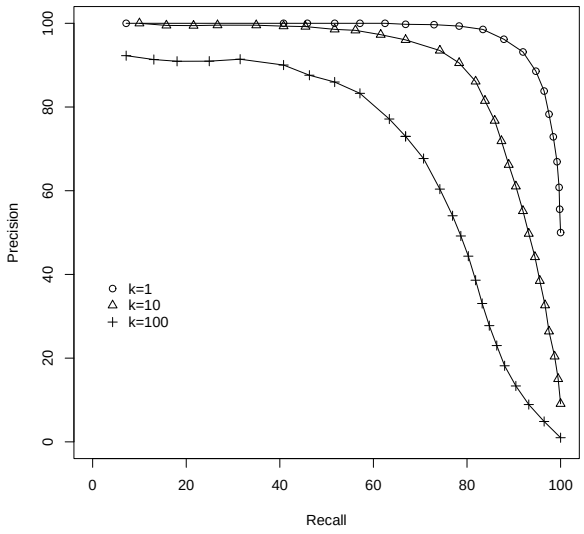


Figure 2: Precision-recall curves for the "artificial intelligence" dataset with different class priors. k is the ratio of the number of negative to the number of positive examples used at testing.

In application to citation recommendation, when only a few citations need to be recommended, we should care only about the high precision mode performance. In the case of the precision recall curve for k = 100, for example, 10% of citations can be recalled for recommendation with 91% precision. This is an overall measure of performance—some users can receive more than enough recommendations, and others none. When we want to recommend a fixed number of citations to *every* user, the CROC performance metric should be used [Schein *et al.*, 2002].

5 Related Work

Integrating or extending existing models or techniques to relational data has been addressed by researchers in several fields, including inductive logic programming, belief nets and link analysis.

A number of approaches extending one-table learners to multi-table domains have been proposed in the inductive logic programming (ILP) community. Generally, these approaches extend learners most suitable to purely binary attributes. Tilde [Blockeel and Raedt, 1998] and WARMR [Dehaspe and Toivonen, 1999], for example, extend decision trees and association rules, respectively. Another ILP approach is *propositionalization*. It uses bodies of first-order rules learned by an ILP technique as binary features in attribute-value learners. Kramer *et al.* [2001] review this methodology. Propositionalization with linear regression modeling, for example, was used by [Srinivasan and King, 1999] to build predictive models in a chemical domain. Decoupling the process of generating features by propositionalization and modeling using these features, however, retains the inductive bias of the technique used to construct features. As one solution, the gener-

ation of binary first-order features by an ILP-style search and feature selection with native criteria of a modeling technique are coupled into a single loop [Blockeel and Dehaspe, 2000; Popescul *et al.*, 2002], e.g. while modeling with a stepwise logistic regression. *Stochastic Logic Programs* [Muggleton, 1995] model uncertainly from within the ILP framework by providing logic theories with a probability distribution.

A number of probabilistic network relational models have also been proposed. *Probabilistic Relational Models* (PRMs) [Getoor *et al.*, 2001] are a relational version of Bayesian networks. PRMs are generative models of joint probability distribution capturing probabilistic influences between entities and their attributes in a relational domain. PRMs present a very powerful formalism. Being a joint probability model of the entire domain, PRMs can provide answers to a large number of possible questions about the domain, including class labels, latent groupings, changing beliefs given new observations. An important limitation, however, of generative modeling is that in reality there is rarely enough data to reliably estimate the entire model. One can achieve superior performance when focusing only on a particular question, e.g. class label prediction, and training models discriminatively to answer that question. A formulation similar to PRMs, but semantically different, called a *Statistical Relational Model*—a statistical model of a particular database instantiation—was proposed for optimizing the answering of relational database queries [Getoor *et al.*, 2002]. Taskar et al. [2002] propose a framework called *Relational Markov Networks* (RMNs)—a relational extension of Markov networks, trained discriminatively following the approach of Lafferty *et al.* [2001]. Here, the structure of a learning domain, determining which relational interactions are explored, is prespecified by a template expressed in a relational query language.

A technique called *Statistical Predicate Invention* [Craven and Slattery, 2001] combines statistical and relational learning by using classifications produced by Naive Bayes as predicates in FOIL [Quinlan and Cameron-Jones, 1995]. Statistical Predicate Invention preserves FOIL as the central modeling component and calls statistical modeling from within the inner structure navigation loop to supply new predicates. Neville and Jensen [2000] propose an iterative technique based on a Bayesian classifier that uses high confidence inferences to improve class inferences for related objects at later iterations. Cohn and Hofmann [2001] propose a joint probabilistic model of document content and connectivity, and apply it to classification tasks, including link prediction. A relational formulation of Markov chains for sequence modeling in web navigation is proposed in [Anderson *et al.*, 2002].

Link analysis plays an important role in the hypertext domains, a notable example being Google, which uses the link structure of the Web by employing a link based concept of page authority in ranking search results [Page *et al.*, 1998]. In addition to knowing the authoritative documents, it is often useful to know the web pages which point to authorities on a topic, the so called called "hub" pages [Kleinberg, 1999], which correspond to the concept of review papers in the scientific literature domain.

6 Discussion and Future Work

We presented the application of statistical relational learning to link prediction in the domain of scientific literature citations. The link prediction task is inherently relational. The noise in the available data sources suggests the use of statistical modeling. *Standard* statistical models, however, usually assume one table domain representation, which is inadequate for this task. Our approach overcomes this limitation. Statistical modeling and feature selection are integrated into a search over the space of database queries generating feature candidates involving complex interactions between objects in a given relational database. This avoids manual feature "packaging" into one table, a process that can be expensive and difficult.

Our method extends beyond ILP because statistics allows generation of richer features, better control of search of the feature space, and more accurate modeling in the presence of noise. On the other hand, our method differs from relational probabilistic network models, such as PRMs and RMNs, because these network models while being able to handle uncertainly, do not attempt to learn and model new complex relationships.

In addition to prediction, learned models can be used for explanatory purposes. Selected features provide insights into the nature of citations. Other linked environments, such as the Web, social networks or biological interactions, we believe, can be explored with the methodology presented in this paper.

We plan to use intelligent search heuristics to speed up the discovery of subspaces with more useful features. Since the potential search space is infinite, intelligent search is necessary to focus the process into more promising subspaces. As one approach, we propose using statistical estimates of "promise" computed by sampling from the subspaces of the same type to decide if those subspaces should be explored more thoroughly.

Using clustering or latent class modeling in statistical relational learning should also prove highly beneficial. Clusters can generate rich relational structure [Foster and Ungar, 2002]. For example, a document belongs to one or more topics. Each of these topics, in turn, has automatically generated properties such as "most frequently cited paper on this topic". Thus a feature such as `most-cited-doc(main-topic(doc-231))` could be learned, as could features involving sets of most cited documents. This has the potential to produce extremely rich and powerful models, helping to overcome problems of data sparsity.

Acknowledgments

The authors would like to thank Steve Lawrence and David Pennock for useful discussions. Small portions of the text also appear in [Popescul *et al.*, 2003].

References

[Anderson *et al.*, 2002] Corin Anderson, Pedro Domingos, and Dan Weld. Relational Markov models and their application to adaptive web navigation. In *Proceedings of the Eighth International Conference on Knowledge Discovery*

and Data Mining (KDD-2002), pages 143–152, Edmonton, Canada, 2002. ACM Press.

[Blockeel and Dehaspe, 2000] Hendrik Blockeel and Luc Dehaspe. Cumulativity as inductive bias. In P. Brazdil and A. Jorge, editors, *Workshops: Data Mining, Decision Support, Meta-Learning and ILP at PKDD-2000*, pages 61–70, 2000.

[Blockeel and Raedt, 1998] Hendrik Blockeel and Luc De Raedt. Top-down induction of logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[Cohn and Hofmann, 2001] David Cohn and Thomas Hofmann. The missing link - A probabilistic model of document content and hypertext connectivity. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2001.

[Craven and Slattery, 2001] M. Craven and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43(1/2):97–119, 2001.

[Dehaspe and Toivonen, 1999] L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.

[Dzeroski and Lavrac, 2001] Saso Dzeroski and Nada Lavrac, editors. *Relational Data Mining*. Springer-Varlag, 2001. ISBN 3540422897.

[Foster and Ungar, 2002] Dean Foster and Lyle Ungar. A proposal for learning by ontological leaps. In *Proceedings of Snowbird Learning Conference*, Snowbird, Utah, 2002.

[Getoor *et al.*, 2001] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*. Springer-Verlag, 2001.

[Getoor *et al.*, 2002] Lise Getoor, Daphne Koller, and Benjamin Taskar. Statistical models for relational data. In *Proceedings of the Workshop on Multi-Relational Data Mining at KDD-2002*, pages 36–55, Edmonton, Canada, 2002.

[Hoff, 2003] P.D. Hoff. Random effects models for network data. In *Proceedings of the National Academy of Sciences: Symposium on Social Network Analysis for National Security*, 2003.

[Hosmer and Lemeshow, 1989] D.W. Hosmer and S. Lemeshow. *Applied logistic regression*. Wiley, New York, 1989.

[Jensen and Neville, 2002] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, pages 259–266. Morgan Kaufmann, 2002.

[Kleinberg, 1999] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[Kramer *et al.*, 2001] S. Kramer, N. Lavrac, and P. Flach. Propositionalization approaches to relational data mining. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 262–291. Springer-Verlag, 2001.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML01*, 2001.

[Lawrence *et al.*, 1999] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[Muggleton, 1995] S. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Proceedings of the 5th International Workshop on Inductive Logic Programming*. Department of Computer Science, Katholieke Universiteit Leuven, 1995.

[Neville and Jensen, 2000] J. Neville and D. Jensen. Iterative classification in relational data. In *Proceedings of AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20. AAAI Press, 2000.

[Page *et al.*, 1998] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the Web. Technical report, Computer Science Department, Stanford University, 1998.

[Plotkin, 1969] G. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 153–163. Edinburgh University Press, 1969.

[Popescul *et al.*, 2002] Alexandrin Popescul, Lyle H. Ungar, Steve Lawrence, and David M. Pennock. Towards structural logistic regression: Combining relational and statistical learning. In *Proceedings of the Workshop on Multi-Relational Data Mining at KDD-2002*, pages 130–141, Edmonton, Canada, 2002.

[Popescul *et al.*, 2003] Alexandrin Popescul, Lyle H. Ungar, Steve Lawrence, and David M. Pennock. Statistical relational learning for document mining. Computer and Information Sciences, University of Pennsylvania, 2003. $http://www.cis.upenn.edu/\tilde{}popescul/publications.html$.

[Quinlan and Cameron-Jones, 1995] J.R. Quinlan and R.M. Cameron-Jones. Induction of logic programs: FOIL and related systems. *New Generation Computing*, 13:287–312, 1995.

[Schein *et al.*, 2002] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25'th International ACM Conference on Research and Development in Information Retreival (SIGIR 2002)*, 2002.

[Schwartz, 1979] Gideon Schwartz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1979.

[Shapiro, 1983] E. Shapiro. *Algorithmic Program Debugging*. MIT Press, 1983.

[Srinivasan and King, 1999] A. Srinivasan and R. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. *Data Mining and Knowledge Discovery*, 3(1):37–57, 1999.

[Taskar *et al.*, 2002] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, Edmonton, Canada, 2002.

Relational Learning Problems and Simple Models

Foster Provost
NYU Stern School of Business
44 West 4th Street
New York, NY 10012, U.S.A.
fprovost@stern.nyu.edu

Claudia Perlich
NYU Stern School of Business
44 West 4th Street
New York, NY 10012, U.S.A.
cperlich@stern.nyu.edu

Sofus A. Macskassy
NYU Stern School of Business
44 West 4th Street
New York, NY 10012, U.S.A.
smacskas@stern.nyu.edu

In recent years, we have seen remarkable advances in algorithms for relational learning, especially statistically based algorithms. These algorithms have been developed in a wide variety of different research fields and problem settings. It is important scientifically to understand the strengths, weaknesses, and applicability of the various methods. However, we are stymied by a lack of a common framework for characterizing relational learning.

What are the dimensions along which relational learning problems and potential solutions should be characterized? Jensen (1998) outlined dimensions that are applicable to relational learning, including various measures of size, interconnectivity and variety; items to be characterized include the data, the (true) model, the background knowledge, and so on. Additionally, individual research papers will characterize aspects of relational learning that they are considering and are ignoring. However, there are few studies or even position papers that examine various methods, contrasting them along common dimensions (one notable exception being the paper by Jensen and Neville (2002b)).

It also is not clear whether straightforward measures of size, interconnectivity, or variety will be the best dimensions. In this paper we argue that other sorts of dimensions are at least as important. In particular, the aforementioned dimensions characterize the learning problem (i.e., the training data and the true model). Equally important are characteristics of the context for using the learned model—which have important implications for learning. For illustration, let us discuss three context characteristics, and their implications for studying relational learning algorithms.

i) Training data versus background knowledge. Relational learning incorporates background knowledge in a more principled manner than is possible with traditional, single-table learning. Here we consider one particular type of background knowledge: descriptions of known objects, of the same type as those to be classified. Dealing with such objects is not an issue for typical (propositional) learning approaches, because data items are assumed to be independent. In contrast, relational models may be able to take advantage of relations between the data to be classified and background-knowledge entities. But what distinguishes background knowledge from training data? Comparison of methods is difficult if they incorporate different assumptions.

Consider the simple question:

Q1: *Should it be possible for the learned model to take advantage of links to specific training entities?*[1]

There is not a simple answer; it depends on the application to which the learning is being applied. This is not a new observation, but to our knowledge it is not dealt with uniformly by researchers and their methods (which makes comparisons difficult).

Let us clarify by defining two, possibly overlapping, sets of data: training data (T) and background knowledge (B). Only B will be available when the model is in use. Q1 then becomes: Is $T \subseteq B$? And if we define $T' = T - (T \cap B)$, then models should *not* be allowed to consider links to entities in T′. This is a simple example, but it is not trivial—which we will discuss more below.

1 To be clear, we are focusing on the links to specific entities themselves, rather than to properties of the entities.

Once such a dimension is agreed upon, we can define mutually acceptable comparative studies, for example that vary the size of T and of B, their overlap, and so on. We also can discuss whether all of B is present at training time, or if new background knowledge may become available when the model is used.

ii) Base-level learning vs. learning determinations. Assume that models may refer only to background-knowledge entities. The question still remains, should they? For example,[2] is it useful to learn that Hitchcock directs horror movies if Hitchcock is dead and is not going to direct any more movies? Wouldn't it be better to learn that `director` strongly determines `genre`?

Q2 (Q1′): *Should it be possible for the model to take advantage of links to specific background entities?*

Again, this depends on the application, and in particular, on the level of generalization desired. Is it appropriate to learn base-level models or higher-level models (or both)? Such higher-level learning has been called learning *determinations* (Russell, 1986; Schlimmer, 1993). For example, in a traditional, single-table setting one may learn that Mexicans speak Spanish and Brazilians speak Portuguese, or at a higher level learn that `Country-of-origin` determines `Language`. A determination is a higher-order regularity that, once known, can be used in a completely new context for learning from very few data, for analogical reasoning, etc.

If the application of the model is going to be in a completely different context (new entities are not linked to previous background knowledge), it may be appropriate to learn higher-level regularities. If the model is going to be used in the same or similar contexts (e.g., it might encounter more Mexicans), base-level learning may be quite appropriate. If the context is uncertain, it may be appropriate to learn both.

iii) Linking to the target values. In traditional flat-table learning, supervised induction *algorithms* reference the target values in the training set. However, it does not make sense for a learned propositional *model* to reference the target values of other examples (because they are assumed to be i.i.d.). However, relational learning does not assume independent entities, and in fact tries to take advantage of linkages between entities. The target values of the linked entities can be treated in different ways, and again, if methods have different treatments comparison is difficult.

[2] Thanks to David Jensen for the example.

Consider another simple question:

Q3: *Should it be possible for the model to consider the target values of linked entities?*

Again, there is not a simple answer: it depends on the application. This question is related to the question of training data versus background knowledge. By our previous discussion, the model should not be able to access the target values of entities in T′. (The induction algorithm, of course, will access these.)

Issues i through iii illustrate dimensions that are qualitatively different from the size, connectivity, homogeneity, etc., of the data or models. These are characteristics of the application that influence what sort of modeling should be done.

Example 1. We want build a model to predict the box-office receipts of movies, using data such as those represented in the Internet Movie Database (IMDB) (Jensen & Neville, 2002a). How do we answer our three questions? We assert that the answer to all three should be "yes." We should not forget about prior movies. (And who knows, a long-lost Hitchcock movie may resurface.) We should not ignore the box-office receipts of prior movies. !

Example 2. We want to predict the subtopic of published academic papers within the area of machine learning, based on their relationships to each other through citations or common authors (McCallum et al., 2000; Taskar et al., 2001). How do we answer our three questions? Again, it depends on the application context. If the models are to be applied to the same area (machine learning) from which the training data were selected, the answer to all three should be "yes." We should not forget about other papers we know about. We should not ignore these papers' subtopics. !

We believe that these three questions will be answered in the affirmative for many applications of relational learning.[3] This has implications for the design and evaluation of relational learning algorithms. Here we discuss only one implication: the answers to these questions may bear on the baseline classification procedure to which other methods are compared. If it is possible for a model to reference the class values of training/background entities, very simple models may perform well.

[3] The answer to one or more questions of course also may be negative. For example, consider classifying web pages from a site different from that used for training (Slattery & Mitchell, 2000).

Consider the following two closely related relational classifiers. Both perform simple combinations of evidence from an entity's relational neighbors. More specifically, these classifiers take advantage of relational "homophily"—the tendency of entities to be related to other similar entities, a form of relational autocorrelation (Jensen and Neville 2002c). The difference between the two classifiers is whether they take advantage of the class labels of linked entities or of explicit class-membership probabilities of linked entities.

Definition. The *degree-k relational-neighbor classifier (k-RN)* estimates the probability of an entity *e* being a member of class *i* as the (weighted) proportion of the background entities linked to *e* by paths of length *k*, that belong to class *i*. !

Definition. The *degree-k probabilistic relational-neighbor classifier (k-pRN)* estimates the class-membership probability of an entity *e* as the normalized sum of the class-membership probabilities of the background (B = T) entities to which *e* is linked by paths of length *k*. !

For a domain of company affiliation classification (Bernstein et al., 2003), for high-autocorrelation affiliations a 1-RN model performs remarkably well—as well as a complicated (and much slower) multi-document text-classification procedure devised specifically for this application, and better than methods based on correlations in stock performance. It clearly would be a mistake to omit this simple model from a comparison of learning techniques for this domain. A k-RN model also works quite well (compared to other methods) for classifying initial public offerings (Perlich & Provost, 2003).

We suggest using a simple, homophily-based classifier (such as one of these relational neighbor classifiers) as a baseline because homophily is ubiquitous in relational data. Jensen & Neville (2002c) found high relational autocorrelation for almost all attributes they examined in linked movie data. Homophily-based classification is one of the primary techniques used in fraud detection [Fawcett & Provost, 1997; Cortes et al., 2001]. Chakrabarti et al. (1998) take advantage of homophily to classify hypertext documents. Furthermore, homophily with respect to a wide variety of descriptive variables has been observed in the interpersonal relationships that define social groups, and is one of the basic premises of theories of social structure (Blau, 1977).

Since we borrowed our two examples from the existing literature, we can ask how the simple relational classifiers perform in comparison to more-complex methods.

Example 1, revisited. Neville et al. (2002) learn models for predicting (inter alia) box-office receipts of movies, using Relational Probability Trees (RPTs) and Relational Bayesian Classifiers (RBCs). The models estimate the probability that a movie "will be" a blockbuster (the box-office receipts exceed $2million). Neville et al. found areas under the ROC curve (AUCs) of 0.82 and 0.85 for RPTs and RBCs, respectively, using a set of eight attributes on related entities, such as the most prevalent genre of the movie's studio.

How well does a homophily-based classifier perform on this problem? Consider 2-RN based on a particular link type (call it $2\text{-RN}_{<\text{link type}>}$). Links between movies are through various other entities (actors, studios, production companies, etc.), and we consider the links to be typed by the entity through which they pass (e.g., $2\text{RN}_{\text{producer}}$ means: how often does the producer produce blockbusters). The relational-neighbor classifiers achieved AUCs of 0.78 and 0.79 for $\text{NumLinks}_{\text{production-company}}$ for $2\text{-RN}_{\text{producer}}$ (respectively). Simply averaging the homophily scores for the various links achieves AUC = 0.82.[4]

Let us pretend that the experimental designs are completely comparable, and ask: Do the more complex models produced by the relational learners perform substantially better than simple classifiers? Before answering this we first must agree on issues i—iii above.

Example 2, revisited. Taskar et al. (2001) learn Probabilistic Relational Models (PRMs) for classifying academic papers within machine learning into one of seven possible subtopics. Figure 1 shows the accuracy of the PRM as larger proportions of the data are used as training data and labeled background knowledge (here, T=B). They varied the proportion of known classes in 10% increments, performing 5-fold cross-validation.

How well does a simple homophily-based classifier perform on this problem? With a moderate amount of labeled background papers, a 1-pRN model performs remarkably well—as well as the PRM. Figure 1 compares the classification accuracy of 1-pRN with the

[4] Learning a linear combination of $2\text{-RN}_{<>}$ for several link types (actor, director, producer, production company) and the number of links (as suggested by David Jensen) of several link types achieves AUC=0.85. Although this does involve some learning, and perhaps a quirk of data entry, it nevertheless only uses a single attribute (the class value) from related entities.

reported results from the PRM. Specifically, using the same data as the prior study, we varied the proportion of papers for which the class initially is known from 10% to 60%, in 5% increments. We performed a 10-fold cross-validation at each setting. For classification, unknown classes of related papers were taken to be the class prior (as defined by the known classes). As shown in Figure 1, although 1-pRN has accuracy of only 50% initially, the accuracy is comparable to that of the PRM once half of the papers are labeled.

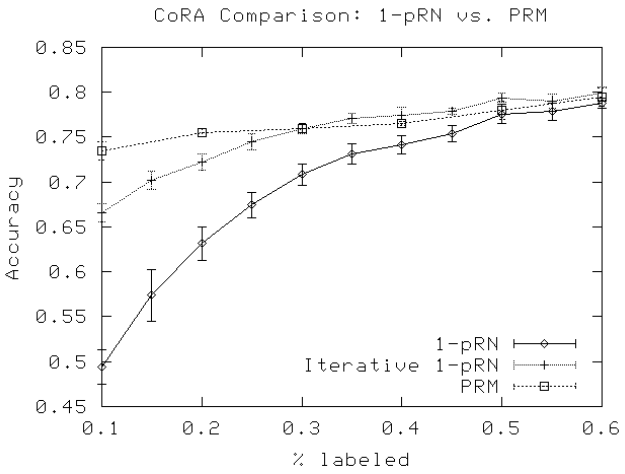


Figure 1: Probabilistic 1-pRN vs. PRM

The poor performance when few of the classes are known is not surprising, as there is little evidence on which the relational neighbor classifier can base its prediction. The PRM uses a form of belief propagation from labeled to unlabeled entities, which may account for the high performance even with only 10% of the entities labeled. Also shown in Figure 1, iterative 1-pRN adds simple belief propagation to the relational-neighbor classifier. Specifically, it estimates the classes of the unknown papers by repeatedly updating the class-probabilities of each initially unknown node (using 1-pRN) until some stopping criterion is met (in this case, we simply let it run for 100 iterations). While there is little difference from 1-pRN when a large percentage of class labels are known initially, iterative 1-pRN shows a marked improvement when fewer class labels are known. In fact, it is quite competitive with the PRM.

Again, if we assume that the experimental designs are comparable, these results provide additional evidence that for this application context (i.e., "yes" to all three questions) such simple models should receive attention.

Final words

In summary, we have argued (1) that we should come to an agreement on dimensions such as these for characterizing relational learning tasks, and (2) that we should be aware of the power of simple models when certain problem formulations are chosen. In particular, we advocate use of the (homophily-based) Relational Neighbor classifiers as baselines for evaluating generalization performance.

Acknowledgements

We thank David Jensen for many discussions that have clarified our thinking on these issues, Jennifer Neville for providing us with the details of their study (and, several years ago, the initial impetus for thinking about accessing the target values), and Ben Taskar and Andrew McCallum for providing us with versions of the Cora data set. Avi Bernstein, Scott Clearwater, and Shawndra Hill worked closely with us on the development and study of kRN and closely related models. This work is sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement F30602-01-2-585.

References

[Bernstein et al., 2003] A. Bernstein, S. Clearwater, and F. Provost. The relational vector-space model and industry classification. *IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, 2003. (Working paper *CDeR #IS-03-02*, Stern School of Business, New York University.)

[Blau, 1977] P. Blau. *Inequality and Heterogeneity: A Primitive Theory of Social Structure.* New York: The Free Press, 1977.

[Cortes et al., 2001] C. Cortes, D. Pregibon, and C. Volinsky. Communities of interest. *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis (IDA-2001)*, pp. 105—114.

[Fawcett & Provost, 1997] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery* 1 (3):291—316, 1997.

[Jensen, 1998] D. Jensen. *Quantitative criteria to characterize KD-ML research for C-XNT*, 1998.

[Jensen and Neville, 2002a] D. Jensen J. and Neville. Data mining in social networks. Invited presentation to the National Academy of Sciences Workshop on Dynamic Social Network Modeling and Analysis. Washington, DC. November 7-9, 2002.

[Jensen and Neville, 2002b] D. Jensen and J. Neville. Schemas and models. In S. Dzeroski, L. De Raedt, and S. Wrobel, editors, *Proceedings of the Workshop on Multi-Relational Data Mining* (MRDM-2002), pages 56—70.

[Jensen and Neville, 2002c] D. Jensen and J. Neville, "Linkage and autocorrelation cause feature selection bias in relational learning," In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[McCallum et al., 2000] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of Internet portals with machine learning. *Information Retrieval*, 3(2): 127—163, 2000.

[Neville et al., 2002] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. University of Massachusetts, (Computer Science Department) *Technical Report 02-55*. Revised Feb. 2003.

[Perlich and Provost, 2003] C. Perlich and F. Provost. Aggregation-based feature invention and relational concept classes. Working paper *CDeR #IS-03-03*, Stern School of Business, New York University, 2003. *To appear in KDD-2003*.

[Russel, 1986] S. J. Russell. Preliminary steps toward the automation of induction. In *Proceedings of the National Conference on Articial Intelligence*, pp. 477—484, 1986.

[Schlimmer, 1993] J. C. Schlimmer. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 284—290, 1993.

[Slattery and Mitchell, 2000] S. Slattery and T. Mitchell. Discovering test set regularities in relational domains. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pp. 895—902, 2000.

[Taskar et al. 2001] B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pp. 870—878.

A Comparison of Stochastic Logic Programs and Bayesian Logic Programs

Aymeric Puech and Stephen Muggleton
Department of Computing, Imperial College London
180 Queen's Gate, London SW7 2BZ, UK
{atp02,shm}@doc.ic.ac.uk

Abstract

First-order probabilistic models are recognized as efficient frameworks to represent several real-world problems: they combine the expressive power of first-order logic, which serves as a knowledge representation language, and the capability to model uncertainty with probabilities. Among existing models, it is usual to distinguish the *domain-frequency* approach from the *possible-worlds* approach.

Bayesian logic programs (BLPs, a *possible-worlds* approach) and stochastic logic programs (SLPs, a *domain-frequency* approach) are promising probabilistic logic models in their categories.

This paper is aimed at comparing the respective expressive power of these frameworks. We demonstrate relations between SLPs' and BLPs' semantics, and argue that SLPs can encode the same knowledge as a subclass of BLPs. We introduce *extended* SLPs which lift the latter result to any BLP. Converse properties are reviewed, and we show how BLPs can define the same semantics as complete, range-restricted SLPs. Algorithms that translate BLPs into SLPs (and vice versa) are provided, as well as worked examples of the inter-translations of SLPs and BLPs.

1 Introduction

The recent study APrIL [Muggleton and de Raedt, 2001] aimed at assessing how probabilistic reasoning could be integrated with first-order logic representations and machine learning. The project ended in June 2002 and raised the conclusion that, *due to the requirements for the use in functional genomics, stochastic logic programs (SLPs) and Bayesian logic programs (BLPs) [were] the most promising formalisms. They represent the most expressive frameworks as they allow not only for constant and predicate symbols but also for functors. Moreover, both formalisms are alternative and complementary formalisms.* Indeed, in BLPs the possible-worlds perspective is dominant, while SLPs use a domain-frequency approach.

1.1 Motivations

Comparing these two major approaches looks like an interesting issue. The study of the relations between the *possible-worlds* and *domain-frequency* approaches, which is not new, crops up in many articles. Halpern dedicated a paper to that study [Halpern, 1989]. Furthermore, the idea of investigating the relations between BLPs and SLPs in particular has been put forward in [Kersting and Raedt, 2000] by Kersting and de Raedt, who reckon at the very end of their article: *Exploring the relations between Bayesian logic programs and stochastic logic programs is interesting because of: (1) both are using SLD trees and (2) transformations between probabilities on the domain and probabilities on possible worlds exist as Halpern noted.*

Essentially two questions at two different levels can be asked:

- Given an expert domain, can we encode the same knowledge in a BLP and in a SLP?
- More practically: can we translate a BLP into an SLP, and an SLP into a BLP? What are the respective interests of these formulations? In particular, does the translation have the same features (computation of the probabilities, inference with or without evidence)?

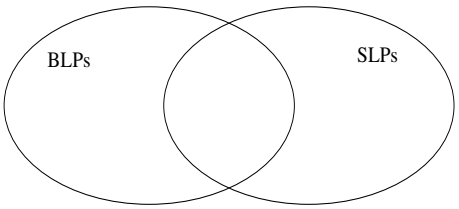


Figure 1: BLPs, SLPs... What is at the intersection?

We argue that it is possible to encode a complete, range-restricted SLP with a BLP, and that the converse holds for a certain subclass of BLPs (the *restricted* BLPs, whose predicate definitions contain a single clause). This latter result is lifted to all BLPs by introducing *extended*

SLPs, which are SLPs augmented with combining functions, and whose evaluation is made in stochastic SLD *and-or* trees instead of stochastic SLD-trees for SLPs.

1.2 Outline

The section 2 of this article is dedicated to some prerequisites (we recall the usual syntax and semantics of SLPs and BLPs). We also demonstrate relations between SLPs' and BLPs' semantics, hence we eventually reduce the issue of inter-translations of SLPs and BLPs to the practical question: given a BLP (resp. an SLP), how can we find an SLP (resp. a BLP) that encodes equivalent distributions of probabilities with respect to the semantics?

The section 3 presents some ways to construct and evaluate an SLP which computes equivalent probabilities to a given BLP.

- Firstly, a *standard translation* is proposed, which deals with the subclass of restricted BLPs, whose *and-or trees* (as defined in [Kersting and Raedt, 2000]) contain only *and*-nodes (these BLPs don't make use of combining rules). *Extended* SLPs are introduced, and the latter result is lifted to all BLPs.
- However, standard translations can only be queried without evidence. Thus a *BN translation* is also proposed, which can be queried with evidence but works only for BLPs with finite Herbrandt model (which obviously prevent the use of functors). Finally, we show how this result can be lifted to all BLPs, provided that we insert a KBMC[1]-inspired stage in the query-answering procedure.

In section 4, we provide a converse translation (from SLP to BLP) and prove that it computes equivalent distributions of probabilities.

We conclude that extended SLPs and BLPs can encode the same knowledge (although their formalism is more or less intuitive, depending on the kind of knowledge we want to model).

2 Background

2.1 Stochastic Logic Programs (SLP)

Stochastic logic programs were first introduced by Stephen Muggleton in 1996, as a generalization of stochastic grammars.

Syntax:

As defined in [Muggleton, 2001], a SLP consists of a set of labelled clauses $p : C$, where p is from the interval $[0, 1]$, and C is a range-restricted[2] definite clause. Later in this report, the labelled clauses $p : C$ will be named *parameterized* clauses or *stochastic* clauses. The simplest example of SLP is the coin example which mimics the action of a fair coin. The probability of the coin coming up either head-side up (0) or tail-side up (1) is 0.5:

$0.5 : coin(0) \leftarrow$
$0.5 : coin(1) \leftarrow$

In [Muggleton, 2001], SLP definition requires that for each predicate symbol q, the probability labels for all clauses with q in the head sum to 1. However, this can be a restrictive definition of SLPs. In other articles ([Cussens, 2000] for instance), SLPs having this property are called ***complete*** SLPs, while in ***uncomplete*** SLPs, the probability labels for all clauses with a same predicate symbol in the head sum to less than 1. In [Cussens, 2000] James Cussens introduces ***pure*** SLPs, whose clauses are all parameterized (whereas ***impure*** SLPs can have non-parameterized clauses. Furthermore, ***normalized*** SLPs are like complete SLPs, but in ***unnormalised*** SLPs, the probability labels for all clauses with a same predicate symbol in the head can sum to any positive value other than 1.

Semantics:

A stochastic logic program P has a distributional semantics, that is one which assigns a probability distribution to the atoms of each predicate in the Herbrand base of the clauses in P. These probabilities are assigned to atoms according to an SLD-resolution strategy which employs a **stochastic selection rule**.

In [Cussens, 1999b], three different related distributions are defined, over derivations, refutations and atoms. Given an SLP P with n parameterized clauses and a goal G, it is easy to define a ***log-linear probability distribution over the set of derivations***, by considering the function:

$$\psi_\lambda(x) \;=\; e^{\lambda.\nu(x)} \;=\; \prod_{i=1}^{n} {l_i}^{\nu_i(x)}$$

where

- x is a derivation in the set of derivations from the goal G.
- $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n) \in \Re^n$ is a vector of log-parameters where $\lambda_i = log(l_i)$, l_i being the label of the clause i.
- $\nu = (\nu_1, \nu_2, ..., \nu_n) \in \{N \cup \infty\}^n$ is a vector of clause counts s.t. $\nu_i(x)$ is the number of times the ith parameterized clause is used in the derivation x.

The proof that $\psi_\lambda(x)$ is a probability distribution (provided that P is pure and normalized) can be found in [Cussens, 2000]. So far we have defined a probability distribution over all possible derivations, but we are mainly interested in the refutations of the goal G. Now, if we assign the probability 0 to all derivations that are not refutations of the goal G, and if we normalize the remaining probabilities with a normalization factor Z, we obtain the ***probability distribution*** $f_\lambda(r)$ ***over the set of the refutations of G***:

$$f_\lambda(r) \;=\; Z^{-1}_{\lambda,G}\; e^{\lambda.\nu(r)}$$

[1] Knowledge-Based Model Construction, as defined in [Kersting and Raedt, 2000].

[2] C is said to be range-restricted iff every variable in the head of C is found in the body of C.

Each refutation r involves some bindings along the SLD-tree, which permits finding the computed answer for G using r. The computed answer is the most general instance of the goal G that is refuted by r; it is also named the ***yield atom***. Let us note $X(y)$ the set of refutations which lead to the yield atom y. We can finally define a ***distribution of probabilities over the set of yield atoms***, with the function:

$$p_{\lambda,G}(y) = \sum_{r \in X(y)} f_\lambda(r) = Z^{-1}_{\lambda,G} \sum_{r \in X(y)} \left(\prod_{i=1}^{n} l_i^{\nu_i(r)} \right)$$

Thus it is fairly simple to define probability-distributions over the yield atoms in SLPs: we can use the **stochastic SLD-tree**, which is the SLD-tree in which each vertex (which corresponds to a resolution step) is labelled with the parameter of the clause that is used in the resolution step. Since every refutation of the goal G corresponds in the stochastic SLD-tree to a branch from the root to a leaf, we only have to:
- draw the stochastic SLD tree,
- compute the probability of each refutation by multiplying the labels along the corresponding branch,
- normalize the probabilities, so that they sum to 1,
- associate each refutation to a yield atom, and sum the probabilities of the refutations that lead to the same yield atom.

We can illustrate this method with Cussens' sample SLP:

$0.4 : s(X) \leftarrow p(X), p(X).$
$0.6 : s(X) \leftarrow q(X).$
$0.3 : p(a).$
$0.7 : p(b).$
$0.2 : q(a).$
$0.8 : q(b).$

We take the goal $G = s(X)$; the stochastic SLD tree which derives from the query $:- s(X)$ is:

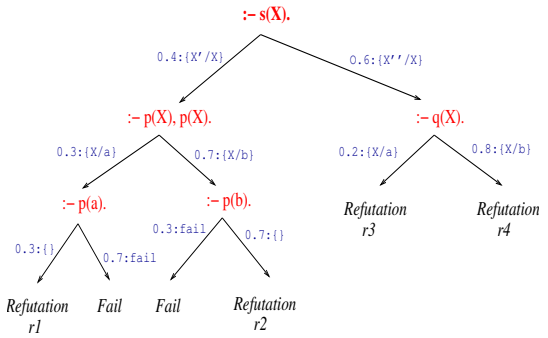


Figure 2: Stochastic SLD tree for the query *?- s(X).*

There are 4 refutations of the goal (r_1 to r_4). The yield atom is $s(a)$ for r_1 and r_3, $s(b)$ for r_2 and r_4. Thus the probability distribution over $\{s(a), s(b)\}$ is $\{Z^{-1} \times (0.4 \times 0.3 \times 0.3 + 0.6 \times 0.2),\ Z^{-1} \times (0.4 \times 0.7 \times 0.7 + 0.6 \times 0.8)\}$ where Z is a normalization constant.

2.2 BLPs

Bayesian logic programs were first introduced by Kersting and De Raedt in 2000, as a generalization of Bayesian nets (BNs) and Logic Programs.

Syntax:

A Bayesian logic program has 2 components: a *logical* one (which is a set of *Bayesian clauses*) and a *quantitative* one (a set of conditional probability distributions and *combining rules* corresponding to that logical structure).

A *Bayesian clause* is an expression of the form:

$$A \mid A_1, ..., A_n$$

where $n \geq 0$ and the A_i are *Bayesian atoms* which are (implicitly) universally quantified. The difference between a logical definite clause and a Bayesian clause is that:
- the sign $\mid$ is employed instead of $:-$,
- Bayesian atoms are assigned a (finite) *domain*, whereas first order logic atoms have binary values.

In order to represent a probabilistic model, we associate with each Bayesian clause c a conditional probability distribution $cpd(c)$ which encodes the probability that $head(c)$ takes some value, given the values of the Bayesian atoms in $body(c)$:

$$\mathbf{P}(head(c) | body(c))$$

This conditional probability distribution is usually represented with a matrix called *conditional probability table* (CPT).

As there can be many clauses with the same head (or non-ground heads that can be unified), we use *combining rules* to obtain the distribution required, i.e. functions which map finite sets of conditional probability distributions onto one *combined* conditional probability distribution. Common combining rules include the *noisy-or* rule, when domains are boolean, and the *max* rule, which is defined on finite domains.

Semantics:

The link to Bayesian networks is now straightforward: each ground Bayesian atom can be associated to a chance node, whose set of states is the domain of the Bayesian atom. The links (influence relations) between chance nodes are given by the Bayesian clauses, and the link matrices by the conditional probability distributions associated to these Bayesian clauses.

The set of ground Bayesian atoms in the least Herbrand model[3] together with the structure defined by the set of ground instances of the Bayesian clauses and the conditional probability tables, define a global (possibly infinite) Bayesian network that can be queried like any other Bayesian net[4].

[3] We define the least Herbrand model of a BLP in the same way as in logic programs.

[4] Bayesian networks are formally defined only for finite sets of chance nodes; this point of view is put forward because it

Thus the query-answering procedure actually consists of two parts: first, given a query and some evidence, the Bayesian network containing all *relevant* atoms is computed, using KBMC (*Knowledge Based Model Construction*). Then the resulting Bayesian network can be queried using any available inference algorithm. Further details about the query-answering procedure can be found in [Kersting and Raedt, 2000].

In the remainder of the paper, we further assume that no merging of nodes takes place in the and-or tree (when constructing the Bayesian net from the and-or tree). It is equivalent to say that the constructed Bayesian net consists of a singly connected network (no loop).

2.3 Formulation of the problem

Halpern's paper [Halpern, 1989] as well as [Cussens, 1999a] are good clarifications about what respective kinds of knowledge can be captured with probabilities on the domain (such as those defined by SLPs) and probabilities on possible worlds (BLPs). Links between these probabilities are also provided.

Let B be a BLP and G_a a ground query. The Bayesian network (BN) constructed with KBMC (as defined in [Kersting and Raedt, 2000]) is denoted by BN_{B,G_a}. The probability of a chance node Q taking the value v in $BN_{B,G}$ (i.e. the probability of the set of possible worlds of $BN_{B,G}$ in which Q has the value v) will be denoted $P_{B,G}(Q = v)$. Given a SLP S, the probability of a ground query G (as defined in the distributional semantics [Muggleton, 2000]) is noted $P(G \ / \ S)$.

The fact that a k-ary Bayesian atom G_a takes the value v can be represented with a $(k+1)$-ary logical atom G having the same predicate and k first arguments as G_a, and the value v as last argument. Conversely, we can identify any logical atom to a Bayesian atom having the domain $\{true, false\}$ and taking the value *true* whenever the logical atom holds.

Hence we will claim that a BLP and an SLP define *equivalent semantics* if the probability that any ground Bayesian atom G_a in the Herbrand model of the BLP takes some value v is identical to the probability of the associated logical atom G in S:

$$P(G \ / \ S) \ = \ P_{B,G_a}(G_a = v)$$

Given these relations between SLPs' and BLPs' semantics, we eventually reduce the issue of inter-translations of SLPs and BLPs to the practical question: given a BLP (resp. an SLP), how can we find an SLP (resp. a BLP) that encodes equivalent distributions of probabilities with respect to the semantics?

3 From BLPs to SLPs

We provide two translations from BLPs to SLPs. The *standard translation* (3.1) actually works for all BLPs but does not handle *evidence* (that is: some prior knowledge about the domain, which corresponds to the instantiation of a chance node in a BN). The reason is that SLPs and e-SLPs define semantics on tree structures, whereas KBMC (in BLPs) permits the union of several trees, hence the computation of probabilities in singly connected networks. Thus we also provide a more global approach with the *BN translation* (3.2).

3.1 Standard Translations

Standard Translation for Restricted BLPs:

Definition: If S is an SLP, the subset S_h of clauses in S with predicate symbol h in the head is called the definition of h. A restricted BLP is a BLP whose predicate definitions contain one single stochastic clause each.

Definition (standard translation of a restricted BLP):

Let B denote a restricted BLP.

- Identify each k-ary Bayesian atom b, which appears in B and has the value domain V, to the $(k+1)$-ary (logical) atom $b(v_b)$ having the same k first arguments and a value v_b of V as last argument.

- For each Bayesian clause $head|b_1, ..., b_n$ in B, for each value in the associated CPT, which indicates the probability $p_{v_h,v_{b1},...,v_{bn}}$ that the Bayesian atom *head* takes the value v_h given that the $\{b_i \ : \ i \in \mathcal{N}_n\}$ take the values $(v_{b1}, ..., v_{bn})$, construct the stochastic clause consisting of the parameter $p_{v_h,v_{b1},...,v_{bn}}$, and the definite clause:

$$head(v_h) \ :- \ b_1(v_{b1}), \ ..., \ b_n(v_{bn})$$

- The standard translation of B consists of the n stochastic clauses constructible in that way, n being the sum of the numbers of coefficients in the CPTs. This SLP is pure and unnormalised (the parameters of the clauses in $S_h \subseteq S$ sum to the product of the domain sizes of the Bayesian atoms in the body of the Bayesian clause with head h).

Theorem: Given a restricted BLP B, its standard translation S obtained as defined above, and a ground Bayesian query G_a. Let us associate to G_a the logical query $G(v)$, $v \in dom(G_a)$. Then: $P(G(v)/S) = P_{B,G_a}(G_a = v)$.

We illustrate the standard translation mechanism through this simple example:

Example:

Let us take the following standard translation of a BLP (the original BLP does not need to be mentioned):

provides a better idea of the relations between BLPs and Bayesian nets.

0.99 : $alarm(A, yes) \leftarrow burglary(A, yes), tornado(A, yes).$
0.80 : $alarm(A, yes) \leftarrow burglary(A, yes), tornado(A, no).$
0.90 : $alarm(A, yes) \leftarrow burglary(A, no), tornado(A, yes).$
0.05 : $alarm(A, yes) \leftarrow burglary(A, no), tornado(A, no).$
0.01 : $alarm(A, no) \leftarrow burglary(A, yes), tornado(A, yes).$
0.20 : $alarm(A, no) \leftarrow burglary(A, yes), tornado(A, no).$
0.10 : $alarm(A, no) \leftarrow burglary(A, no), tornado(A, yes).$
0.95 : $alarm(A, no) \leftarrow burglary(A, no), tornado(A, no).$
0.4 : $burglary(A, yes) \leftarrow neighborhood(A, bad).$
0.2 : $burglary(A, yes) \leftarrow neighborhood(A, avg).$
0.1 : $burglary(A, yes) \leftarrow neighborhood(A, good).$
0.6 : $burglary(A, no) \leftarrow neighborhood(A, bad).$
0.8 : $burglary(A, no) \leftarrow neighborhood(A, avg).$
0.9 : $burglary(A, no) \leftarrow neighborhood(A, good).$
0.3 : $neighborhood(tom, bad).$
0.4 : $neighborhood(tom, avg).$
0.3 : $neighborhood(tom, good).$
0.01 : $tornado(tom, yes).$
0.99 : $tornado(tom, no).$

We want to compute the probability $P(burglary(tom, yes) \ / \ S)$. Each refutation in the stochastic SLD-tree rooted at $burglary(tom, yes)$ permits the computation of the probability of a particular set of possible worlds. The nodes along the refutation correspond to instantiations of some Bayesian atoms. The set of possible worlds we are talking about is the set of worlds where the instantiations defined along the refutation hold.

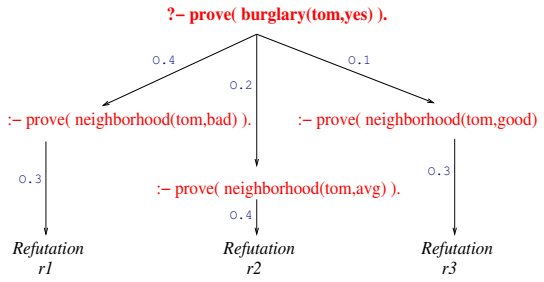


Figure 3: Stochastic SLD tree for $?- burglary(tom, yes)$.

- The refutation 1 is associated to the set of possible worlds in which $\{burglary(tom) = yes, neighborhood(tom) = bad\}$. $Pr = 0.4 \times 0.3 = 0.12$
- The refutation 2 is associated to the set of possible worlds in which $\{burglary(tom) = yes, neighborhood(tom) = avg\}$. $Pr = 0.2 \times 0.4 = 0.08$
- The refutation 3 is associated to the set of possible worlds in which $\{burglary(tom) = yes, neighborhood(tom) = good\}$. $Pr = 0.1 \times 0.3 = 0.03$.

Hence the probability $P(burglary(tom, yes) \ / \ S)$ is equal to $0.12 + 0.08 + 0.03 = 0.23$. In the BN constructed from the original BLP, we have as well: $Pr(burglary(tom, yes)) = Pr(\{possible\ worlds\ where\ burglary(tom) = yes\}) = 0.12 + 0.08 + 0.03 = 0.23$.

Extended SLPs:

Restricted BLPs don't make use of the *or-nodes*, in that all queries can match at most one head. In order to lift the latter result to general BLPs, we have to introduce an extension of SLPs, namely *extended SLPs*.

Definition (Syntax of Extended SLPs): An extended SLP (e-SLP) is an SLP S augmented with a set of *combining functions* CR_h, for all predicates h appearing in the head of some stochastic clause in S. A combining function is a function that maps a set of possible resolvents of h (obtained using one clause in S_h) and associated real numbers in $[0, 1]$ to a real number in $[0, 1]$:

$$CR_h \ : \ ((r_1, p_1), ..., (r_n, p_n)) \ \mapsto \ r \in [0, 1]$$

Definition (Proof of Extended SLPs): Given an e-SLP S_e consisting of the SLP S and the combining functions $(CR_h)_h$, and a query Q (consisting of the predicate p with none or more arguments), the probability $P_e(Q/S_e)$ is the probability of the *pruned* and-or tree T rooted at the or-node Q. The probability of a pruned and-or tree is defined by structural induction:

- Base case: if T is a single or-node, $P_e(Q/S_e)$ is the probability $P(Q/S)$.
- If the root of T is an or-node with n branches leading to the resolvents (and-nodes) $(r_i)_{i \in \mathcal{N}_n}$, then $P_e(Q/S_e) = CR_p((n_i, p_i)_{i \in \mathcal{N}_n})$, where p_i is the probability of the pruned and-or subtree rooted at the and-node r_i.
- If the root of T is an and-node leading to the resolvents (or-nodes) $(r_i)_{i \in \mathcal{N}_n}$, then $P_e(Q/S_e) = \prod_{i=1}^{n} p_i$, where p_i is the probability of the pruned and-or subtree rooted at the or-node r_i.

Standard Translation for BLPs:

Definition (Standard Translation of a BLP): Let B denote a BLP. The standard translation of B is the extended SLP S_e defined by the following stochastic clauses and combining functions:

- The stochastic clauses (which form the set S) are obtained in the same way as the stochastic clauses obtained from a restricted BLP (definition 2).
- Let us take a ground predicate h in the head of some clause in S and assume that it can be unified with the heads of some clauses in S_h, leading to the resolvents $(r_{i,j})_{i,j}$ with probabilities in S equal to $(p_{i,j})_{i,j}$. A resolvent can contain several atoms. The clauses in S_h come from z different Bayesian clauses with the same predicate in the head. These original clauses can be indexed with a number that

corresponds to the first index $i \in \mathcal{N}_z$ in the name of the resolvents. The second index $j \in \mathcal{N}_{n_i}$ refers to different distributions of values over the Bayesian atoms in the body of the Bayesian clause i. We define CR_h by:

$$CR_h = \sum_{j_1 \in \mathcal{N}_{n_1}, \ldots, j_z \in \mathcal{N}_{n_z}} CR(h, r_{1,j_1}, \ldots, r_{z,j_z}) \times \prod_{t=1}^{z} p_{t,j_t}$$

where CR is the combining rule defined in B.

Proposition: The first theorem stating the equivalence of the semantics for the standard translation of restricted BLPs still holds if the translation is done with the latter rules (using e-SLPs and probabilities defined in definition 4).

Theorem: Given any BLP B, its standard translation S_e obtained as defined above, and a ground Bayesian query G_a. Let us associate to G_a the logical query $G(v)$, $v \in dom(G_a)$. Then: $P_e(G(v)/S_e) = P_{B,G_a}(G_a = v)$.

3.2 BN Translations

It is a well-known result that BNs can be formulated in terms of SLPs. The next subsection recalls Cussens' suggestion of encoding. Since Herbrand bases of the BLPs define (possibly infinite) Bayesian networks, a possible way of translating a BLP into an SLP is to encode the corresponding Bayesian net.

From Bayesian Nets to SLPs:

According to Cussens, *unnormalised SLPs can conveniently represent Bayesian nets.*

The encoding is presented throughout an example. Let us take the following Bayesian net.

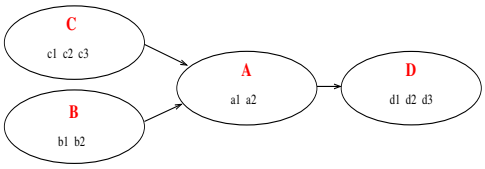


Figure 4: A simple Bayesian net

The following SLP encodes this Bayesian network:

```
1: world(A,B,C,D) :-
       b(B), c(C),
       a(A,B,C),
       d(D,A).

0.05: b(b1).
0.95: b(b2).

0.07: a(a1,b2,c1).
...
```

The first stochastic clause[5] permits the *possible-worlds* approach, since **the probability of the refutation of a ground instance of the predicate symbol *world* is exactly the probability of the associated possible world**.

The next clauses encode the values that are contained in the CPTs. They are unary stochastic clauses *groundterm* associated to the parameter $P(groundatom)$.

From Cussens' point of view the translation from BN to SLP seems obscure in that the directionality of BN is obscured. Indeed, the labels of the clauses in the resulting SLP encode *conditional* probabilities, while the predicate symbols don't seem to make any distinction between the head and the other variables the head depends on. However one can argue that the structure of the BN is encoded in the first stochastic clause, the *query clause* as defined above. This approach works pretty well and will be used in the next section as the basis for the translation of BLPs.

Lifting this approach to BLPs:

The resulting SLP falls into two parts: a *data* part (consisting of several stochastic clauses which encode the knowledge in the conditional probability tables) and a *query* part (consisting of only one stochastic clause with the parameter 1.0).

Data Part

In order to mimic the global approach that was presented in the previous section, we propose to use the following clauses to encode the knowledge in the conditional probability tables of our example:

$$\begin{array}{lcl} 0.99 & : & holds_1(alarm(A, yes), burglary(A, yes), tornado(A, yes)) \\ 0.8 & : & holds_1(alarm(A, yes), burglary(A, yes), tornado(A, no)). \\ \ldots & & \ldots \\ 0.4 & : & holds_2(burglary(A, yes), neighborhood(A, bad)). \\ \ldots & & \ldots \\ 0.99 & : & holds_4(tornado(tom, no)). \end{array}$$

Note that *alarm*, *burglary*... were predicate symbols in the last *standard* translation, while we consider them as functors from now on.

Query Part

Let us recall that a BLP defines a (possibly infinite) Bayesian network whose chance nodes are the atoms in the least Herbrand model associated to the set of Bayesian clauses (this set can indeed be identified to a DCL (Prolog-like) program). Now there are two options:

• **A)** Either **the least Herbrand model HM is finite**: in that case the *query clause* can contain all atoms in HM, and the probability of the refutation of a ground instance of the query clause is exactly the probability of the corresponding possible world.

In the *alarm* example, the Herbrand model HM is finite:$HM = \{alarm(tom), burglary(tom), neighborhood(tom), tornado(tom)\}$.

Thus the query clause will be:

[5]This clause is obviously different from all other clauses in the SLP. We will call it the ***query clause*** of the SLP.

$$
\begin{aligned}
1 \quad : \quad & holds(\\
& \quad tornado(tom, A), \\
& \quad alarm(tom, B), \\
& \quad burglary(tom, C), \\
& \quad neighborhood(tom, D) \ ) \ \leftarrow \\
& holds_3(neighborhood(tom, D)), \\
& holds_2(burglary(tom, C), neighborhood(tom, D)), \\
& holds_4(tornado(tom, A)), \\
& holds_1(alarm(tom, B), burglary(tom, C), \\
& \quad tornado(tom, A).
\end{aligned}
$$

Implementations have been carried out in Prolog, so that it is possible to query the resulting SLP (data part + query clause) by asking Prolog:

$$
?- \ query(\ holds(\ tornado(tom, A), alarm(tom, B), \\
burglary(tom, C), neighborhood(tom, D)\ )\ ).
$$

The variables A, B, C or D can be replaced by constants (*yes*, *no*, *good*, *bad* or *avg* when appropriate), whenever evidence must be taken into account.

• **B)**... Or **the least Herbrand model** HM **is infinite**: then we don't know what a *possible world* will consist of, since it clearly depends on the query. Furthermore, we don't even know what evidence can be declared (the actual chance nodes of the Bayesian net depend on the query). Thus **we need some additional stage to query the SLP**, which can replace the -cumbersome- construction of the Bayesian net with KBMC.

To construct the query clause, we need to have an idea of the corresponding Bayesian net that would be generated in the BLP with the same query. In KBMC, all *relevant* Bayesian atoms are determined by calculating the and-or trees of the query and the evidence, and by merging these trees. We use a slightly different approach: given a query (assimilated to a ground Bayesian atom), we use the structure of the Bayesian clauses to determine: - all *influencers* of the query: this is a kind of *deductive* approach, since the influencers are the atoms that appear in the refutation of the query, when assimilating the BLP with a LP. - all *influenced atoms* of the query: this is a kind of *abductive* approach, since we will look for the Bayesian clauses whose body contain the query.

4 From SLPs to BLPs

In [Kersting and Raedt, 2000], Kersting and de Raedt show that any logic program can be formulated in terms of BLPs: they assign the domain $\{true, false\}$ to every atom in the Herbrand base of the logic program, and associate the naive conditional probability tables to the clauses, which is defined as follows:

- the probability that the head takes the value *true* given that all atoms in the body have the value *true* is 1.0.
- the probability that the head takes the value *true* given any other distribution of values over the atoms in the body is 0.0.

Kersting and de Raedt claim that this BLP (together with the *noisy-or* or the *max* combining rule) mimics the original logic program. How can we lift this translation mechanism to SLPs?

In order to shift the approach from a possible-worlds to a domain-frequency perspective (which is essentially a *single-world* perspective), the idea is to assign non-zero probabilities to *only one* set of values of the body. Here we propose a way to translate into a BLP: the resulting BLP can compute the same probabilities as the distributional semantics defined in [Muggleton, 2001].

Definition (translation of an SLP):
Let P denote a complete, range-restricted and non-recursive SLP.

- For each stochastic clause $p \ : \ head \leftarrow b_1, ..., b_n$ in P, identify each atom to a Bayesian atom whose domain is $\{true, false\}$.
- Construct the Bayesian clause having the same head, the same body, and the following conditional probability table:

			head	
b_1	...	b_n	true	false
true	true	true	p	$1-p$
true	true	false	0	1
≀	≀	≀	0	1
false	false	false	0	1

- To complete the definition of the BLP, we need to define a *combining rule* CR. Suppose that we have to combine n conditional probability tables CPT_i $(1 \leq i \leq n)$. Each CPT_i defines the probabilities $P(head \mid \mathcal{B}_i)$, where $\mathcal{B}_i$ is the set of ground Bayesian atoms in the body of the associated clause. Thus to define $CR(\ (CPT_i)_{1 \leq i \leq n}\ )$, and by using normalization, we only have to set the values of $P(head = true \mid \cup_{i=1}^{n} \mathcal{B}_i)$ for all possible instantiations of the ground Bayesian atoms in $(\cup_{i=1}^{n} \mathcal{B}_i)$. The value of $P(head = false | \cup_{i=1}^{n} \mathcal{B}_i) = 1 - P(head = true | \cup_{i=1}^{n} \mathcal{B}_i)$ can then be deduced.
- For each possible instantiation $(\cup_{i=1}^{n} Inst_i)$ of $(\cup_{i=1}^{n} \mathcal{B}_i)$, we take the **sum** $\sum_{i=1}^{n} P(head = true \mid \mathcal{B}_i = Inst_i)$ and assign it to $P(head = true \mid \cup_{i=1}^{n} \mathcal{B}_i)$. Since the SLP is complete, this sum will never be greater than 1, and the CR is well defined.

Theorem: Given a complete, range-restricted and non-recursive SLP S, its translation into a BLP B obtained as defined above , and a ground query G. Let us associate to G the Bayesian atom G_a, whose domain is $\{true, false\}$, and which is itself associated to a chance node in the Bayesian net BN_{B,G_a}. Then: $P(G/S) = P_{B,G_a}(G_a = true)$.

We will now examine two examples.

Example (nbiased coin): Let us recall the coin example. The SLP is complete, range-restricted and not recursive.

$$
\begin{aligned}
0.5 \quad : \quad & coin(0) \leftarrow \\
0.5 \quad : \quad & coin(1) \leftarrow
\end{aligned}
$$

In the coin problem, the very simple BLP that we construct doesn't make use of any combining rule. It only contains 2 Bayesian clauses. Using the Prolog formalism introduced in the second chapter, the BLP will be written:

```
coin(0).
coin(1).
```

coin(0)	
true	false
0.5	0.5

coin(1)	
true	false
0.5	0.5

If we query this BLP with, for example, "?− $coin(1)$.", the KBMC will result in the Bayesian net containing the single chance node *coin(1)*, whose probability of being *true* will be 0.5 as required by the distributional semantics in [Muggleton, 2001].

We will finally present some more complex example.

Example:

Let P be the complete, range-restricted SLP consisting of the following stochastic clauses:

$$\begin{array}{lcl} 0.4 & : & s(X) \leftarrow p(X), q(X). \\ 0.6 & : & s(X) \leftarrow r(X). \\ 0.3 & : & p(a). \\ 0.7 & : & p(b). \end{array} \qquad \begin{array}{lcl} 0.3 & : & q(a). \\ 0.7 & : & q(b). \\ 0.2 & : & r(a). \\ 0.8 & : & r(b). \end{array}$$

If we follow the method presented above, we obtain the BLP:

```
s(X) | p(X), q(X).          q(a).
s(X) | r(X).                q(b).
p(a).                       r(a).
p(b).                       r(b).
```

		s(X)	
p(X)	q(X)	true	false
true	true	0.4	0.6
true	false	0.0	1.0
false	true	0.0	1.0
false	false	0.0	1.0

	s(X)	
r(X)	true	false
true	0.6	0.4
false	0.0	1.0

p(a)	
true	false
0.3	0.7

The CPTs for p(b), q(a), q(b), r(a) and r(b) are not detailed.

Now, if we query this BLP with "? − $s(a)$.", the resulting Bayesian network will contain 4 chance nodes: $p(a)$, $q(a)$ and $r(a)$ directly influence $s(a)$. The combining rule gives the combined conditional probability table as follows.

			s(a)	
p(a)	q(a)	r(a)	true	false
true	true	true	1.0	0.0
true	true	false	0.4	0.6
true	false	true	0.6	0.4
true	false	false	0.0	1.0
false	true	true	0.6	0.4
false	true	false	0.0	1.0
false	false	true	0.6	0.4
false	false	false	0.0	1.0

By using any standard inference algorithm in this BN (e.g. Pearl's message passing), we obtain the probability $P(s(a) = true) = 0.156$, which is the result we were looking for (since the distributional semantics gives: $P(s(a)\ /\ P) = (0.4 \times 0.3 \times 0.3 + 0.6 \times 0.2) = 0.156$).

5 Conclusion and Further Work

We have demonstrated relations between SLPs' and BLPs' semantics, and we have shown that SLPs augmented with combining functions (namely *extended SLPs*) and BLPs can encode the same knowledge, in that they encode equivalent distributions of probabilities with regards to the latter relations. Since SLPs need to be augmented with combining rules in order to be as expressive as BLPs, and BLPs are able to encode complete, range-restricted and non-recursive SLPs, we are tempted to conclude that BLPs are more expressive than strict SLPs.

However, SLPs' and BLPs' formalisms are more or less intuitive, depending on the kind of knowledge we want to model. It should be noted that BLPs' query-answering procedure is cumbersome because of KBMC and the necessity of using different frameworks (computational logic, Bayesian nets), while inference mechanisms in SLPs are straightforward.

We believe this paper to be a formal basis for several further studies. In the perspective of inductive learning, inter-translations of e-SLPs and BLPs can be used to extend learning techniques designed for BLPs to the learning of e-SLPs (and vice-versa). We think it could be interesting to investigate the interests of such extensions.

So far e-SLPs have been introduced in a fairly general way; the definition of suitable constraints on the combining functions in e-SLPs is also a precondition for their *learnability*.

Acknowledgments

The first author was supported by the Engineering and Physical Sciences Research Council (EPSRC). The second author acknowledges support from the ESPRIT IST project *Application of Probabilistic Inductive Logic Programming* (APRIL, IST-2001-33053). We would like to thank Luc De Raedt and Kristian Kersting for helpful discussions on this topic. We would also like to thank anonymous reviewers for their comments and constructive criticism.

References

[Cussens, 1999a] J. Cussens. Integrating probabilistic and logical reasoning. In *Electronic Transaction on Artificial Intelligence*, 1999. Machine Intelligence Workshop (MI16), special issue, (submitted).

[Cussens, 1999b] James Cussens. Loglinear models for first-order probabilistic reasoning. In Kathryn Blackmond Laskey and Henri Prade, editors, *Proceedings of the 15th Annual Conference on Uncertainty in AI (UAI'99)*, pages 126–133. Morgan Kaufmann, 1999.

[Cussens, 2000] J. Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2000.

[Halpern, 1989] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1989.

[Kersting and Raedt, 2000] Kristian Kersting and Luc De Raedt. Bayesian logic programs. In J. Cussens and A. Frisch, editors, *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pages 138–155, 2000.

[Muggleton and de Raedt, 2001] S. Muggleton and L. de Raedt. *Application of Probabilistic Inductive Logic Programming, Final and Periodic Progress Report*. European Union IST programme Assessment, Project 33053, 2001.

[Muggleton, 2000] S. Muggleton. Learning stochastic logic programs. *Electronic Transactions in Artificial Intelligence*, 4(041), 2000.

[Muggleton, 2001] S. Muggleton. Stochastic logic programs. *Journal of Logic Programming*, 2001. Accepted subject to revision.

Why the Title of This Workshop Should Be "Learning Relational Statistical Models from Data"

Stuart Russell
Computer Science Division
University of California
Berkeley, CA 94720-1776
russell@cs.berkeley.edu

The assumption underlying the title "Learning Statistical Models from Relational Data" is that relational data exist.

Data are actual observations. The observation of some random variable X gives us a value x; procedurally speaking, we obtain x and we assign it (with certainty) as the value of X. We can have certainty in this sense because X may be defined as the random variable whose value will be obtained by the observation procedure (e.g., "the outcome of the next coin flip"). It can, of course, be the case that X corresponds to a noisy measurement of some other quantity W (e.g., X is the "measured battery charge remaining" and W is the "true battery charge remaining"). Again, we typically know the connection between W and X with certainty. Since many observations are noisy, the ability to distinguish true and measured values is important.

When we observe a tuple "`r(a,b)`" (say, in a database), of what random variable is this an observation? Here are some possible answers:

- It is an observation of the Boolean random variable $R(A, B)$. That is, we are uncertain as to whether object A is related by R to object B, and the observation settles the issue with certainty. This is the sense in which relational data exist.
- "`a`" was mistyped, and the observation actually concerns the Boolean random variable $R(C, B)$.
- "`b`" was mistyped, and the observation actually concerns the Boolean random variable $R(A, D)$.
- The data entry clerk was confused about the argument order and the observation actually concerns the Boolean random variable $R(B, A)$.
- "`r`" was mistyped, and the observation actually concerns the Boolean random variable $S(A, B)$.
- Both "`a`" and "`b`" were mistyped, etc., etc.

Hence, unless the model-builder is absolutely certain of the correctness and uniqueness of the identifiers used in the atom—i.e., there is no noise in the observation—it is better to view the observation not as a relational datum, but as an observation of three string-valued random variables. In theory, it could amount to an observation of any of the Boolean random variables corresponding to relations between pairs of objects.

The appropriate level of scepticism depends on the application. Data entry software may be such that confusing the $Supervisor$ relation with the $OfficeNumber$ relation is impossible. The first argument "`a`" may be a long identifier, such that the probability of misidentification is negligible provided the software does not allow assertions about new identifiers. (Note, however, that my social security number is sometimes used by a small printing company in Indiana, thanks to a transposition of two digits.) Even if the identifiers in one database are never confused, however, we may need to merge (or learn models from) two different databases that use different identifiers. In that case, there is often uncertainty as to which identifiers are equivalent—e.g., is the owner of account "9999-999999", whose name is recorded as "Stewart Russell" at "263 Hilcrest", the same person as the owner of account "1234-567890", whose name is recorded as "Stuart J. Russell" at "263 Hillcrest Rd."?

The conflation of tuple observations with relational data has been noticed in the context of Web data. Some projects have viewed the existence of a link between two URLs (with appropriate anchor text) as an observation of a particular relation between objects somehow connected with those URLs. Commentators have observed that this could cause difficulties. The home page of a student may contain "student at `<A HREF="www.stanford.edu">`Palo Alto Junior College`</A>`" but of course the preceding words might be "My chihuahua Tiggy is a" or "There is no truth in the rumor that I was once a".

The discussion above assumes that data are already available in tuple form, whether noisy or not. In many actual applications, data are gathered not from relational databases but from text, Web pages, speech, cameras, instruments, etc. Unique identifiers (other than perhaps for the observation objects themselves, e.g., the URL as an identifier of a Web page (if we neglect time)) are generally unavailable in such data, as are relation names. For example, papers have reference lists that "refer" to other papers, but it may require a good deal of sophisticated probabilistic reasoning to work out which papers those might be. Finally, there is the question of the origin of the "refers" relation. We cannot address this question at all if data are already relational.

Learning Statistical Models of Time-Varying Relational Data

Sumit Sanghai, Pedro Domingos and Daniel Weld
Department of Computer Science and Engineering
University of Washington, Seattle, WA 98195

1 Introduction

Formalisms that can represent objects and relations, as opposed to just variables, have a long history in AI. Recently, significant progress has been made in combining them with a principled treatment of uncertainty. In particular, probabilistic relational models or PRMs [4] are an extension of Bayesian networks that allows reasoning with classes, objects and relations. Although PRMs have been successfully applied to a lot of different domains, they lack the temporal dynamics of the real world. In most real world systems, objects get created, modified and even deleted over time. Similarly, the relationships between objects change as time progresses. For example, consider the problem of predicting the set of research topics that become "hot" (e.g., as measured by the number of papers published about them) over time, the changing distribution of these topics among conferences, and the interests and collaborations between authors. It would be difficult to learn a PRM that modeled this time-varying behavior.

Currently the most powerful representation available for capturing sequential phenomena is dynamic Bayesian networks (DBNs) [1], but DBNs are unable to compactly represent many real-world domains that contain multiple objects and classes of objects, as well as multiple kinds of relations among them. DBNs are even more awkward if one wishes to model objects and relations that appear and disappear over time. Thus, our research has focused on a new representation, *dynamic probabilistic relational models* (DPRMs) which combines PRMs with DBNs. Previously, we have explored the problem of efficient inference [8]; this paper outlines our thoughts on learning DPRMs.

2 Dynamic Probabilistic Relational Models

We start by briefly summarizing the definition of PRMs and DPRMs, adapted from [4; 8]. A PRM encodes a probability distribution over the set of all possible instantiations I of a schema. In the simplest case, the relational attributes of all objects are assumed to be known, and the PRM specifies a probability distribution for each propositional attribute A of each class C. The parents of each attribute (i.e., the variables it depends on) can be other attributes of C, or attributes of classes that are related to C by some slot chain. Thus, by knowing the relational attributes one can get the joint probability distribution by computing the set of parents for each object and its attributes and calculating the probability through the distribution specified. More generally, only the object skeleton might be known, in which case the PRM also needs to specify a distribution over the relational attributes [5].

Now, we extend PRMs to handle the time domain in the same way that DBNs extend Bayesian networks. Given a relational schema $\mathcal{S}$, we first extend each class C with the relational attribute C.*previous*, with domain C. As before, we initially assume that the relational skeleton at each time slice is known.

Definition 1 A *two-time-slice PRM (2TPRM)* for a relational schema $\mathcal{S}$ is defined as follows. For each class C and each propositional attribute $A \in \mathcal{A}(\mathcal{C})$, we have:

- A *set of parents* $Pa(C.A) = \{Pa_1, Pa_2, ..., Pa_l\}$, where each Pa_i has the form $C.B$ or $f(C.\tau.B)$, where τ is a slot chain containing the attribute *previous* at most once, and $f()$ is an aggregation function.
- A *conditional probability model* for $P(C.A|Pa(C.A))$. □

Definition 2 A *dynamic probabilistic relational model (DPRM)* for a relational schema $\mathcal{S}$ is a pair $(M_0, M_{\rightarrow})$, where M_0 is a PRM over I_0, representing the distribution P_0 over the initial instantiation of $\mathcal{S}$, and $M_{\rightarrow}$ is a 2TPRM representing the transition distribution $P(I_t|I_{t-1})$ connecting successive instantiations of $\mathcal{S}$. □

DPRMs are extended to the case where only the object skeleton for each time slice is known in the same way that PRMs are, by adding to Definition 1 a set of parents and conditional probability model for each relational attribute, where the parents can be in the same or the previous time slice. When the object skeleton is not known (e.g., if objects can appear and disappear over time), the 2TPRM includes in addition a Boolean existence variable for each possible object, again with parents from the same or the previous time slice.

3 Inference in DPRMs

Just as a PRM can be expanded into a Bayesian network, so can a DPRM be unrolled into a DBN. In principle, we can then perform inference using particle filtering [2], the most widely used approximate inference algorithm for DBNs. Particle filtering maintains a set of samples (particles) to approximate the distribution of any state; the distribution for next state is achieved by importance sampling and resampling. Unfortunately, for DPRMs, particle filtering is likely

to perform poorly, because the state space will be huge. We overcome this by adapting Rao-Blackwellisation [7] to the relational setting. Rao-Blackwellisation divides the state variables into two sets — one in which values are inferred using a particle filter and the other in which values are calculated analytically from the values of the variables in the first set. We make the major assumption that relational attributes do not appear anywhere in the DPRM as parents of unobserved attributes, and that each reference slot can be occupied by at most one object. Then, a Rao-Blackwellised particle is composed of sampled values for all propositional attributes of all objects, plus a probability vector for each relational attribute of each object which is inferred exactly.

While this technique can vastly reduce the size of the state space which particle filtering needs to sample, storing and updating all the requisite probabilities can still become quite expensive. This expense can be ameliorated if context-specific independences exist. We can then replace the vector of probabilities with a novel tree structure whose leaves represent probabilities for entire sets of objects [8].

Our experiments evaluated the efficiency of several inference schemes applied to an assembly-plan execution monitoring task in a simplified manufacturing domain. Even with hundreds of thousands of particles, standard particle filtering failed (i.e. terminated due to inconsistent observations which could not be explained) on datasets with around 100 objects and 500 time steps. In contrast, our inference algorithm yielded accurate predictions on similar problems with only 5000 particles, and ran more quickly and with less storage [8].

Much work remains to improve inference. For example, we will endeavor to lift the assumptions mentioned above and more effectively use a DPRM's structure during inference.

4 Learning in DPRMs

When a DPRM consists of only a single time slice it becomes equivalent to a PRM, and when the DPRM is devoid of relations it is a DBN. Thus we look to combine the learning algorithms already developed for PRMs and DBNs. The first step, parameter learning, appears to be relatively straightforward when no data is missing, since the parameters associated with different types of nodes can be estimated individually. However, there is a subtlety which makes the problem more complex than in a DBN:

- A DPRM can generate a unique state in multiple ways, and each way must be considered during parameter estimation.

For example, if in the new state objects get created, the order of creation can affect the likelihood of the data, as the newly-created objects can interact with each other. There may be a combinatorial number of ways in which a DPRM may generate each state, so we are developing methods to do parameter estimation efficiently. One possibility is to impose a canonical ordering, and another is to greedily compute the most likely order(s) in which the data could have been generated.

In order to learn the DPRM structure, we have to take care of several more issues:

- Defining constraints to eliminate illegal DPRMs is essential when navigating the space of structures. A cycle in a PRM is illegal, and this constraint extends to the two parts of a DPRM. There are additional constraints on a 2TPRM; specifying these in a way that allows creation of an unbounded number of dynamic objects is challenging.
- There are several strategies for searching the space of DPRM structures. The simplest idea is to add and delete edges in the two components, PRM and 2-TPRM, to generate candidate DPRMs. One could do the search by first learning a PRM which gives a good intra-time-slice connectivity, before learning the inter-time-slice connectivity.
- An important task is scoring a DPRM, e.g. with a likelihood-based measure. To compute the likelihood of the data given a candidate DPRM, fast DPRM inference is required. While our particle filtering algorithm is quite fast, we wish to extend it so that we can efficiently explore the space of DPRMS, incrementally updating the likelihood scores. We believe the two-phase search strategy suggested previously will simplify this task.
- Since the space of candidate DPRM models is huge, we are considering pruning mechanisms. Note that some of the methods stated above actually prune the space (e.g. learning the PRM first, followed by time dependencies). One may also impose priors on the models to bias towards simplicity by limiting the number of edges. We plan to design priors over DPRM structures by extending the approach of Heckerman et al.[6] who exponentially penalize arc differences from a "best" prior structure. We will compare the relative benefits of doing this at the class vs. instance level.
- We plan to extend the learning algorithm to work in the presence of missing values and hidden variables. EM is easiest to apply when the observations are relational but the hidden state is not. Solving this problem with full generality would require an extension of structural EM [3], but this needs to be done for PRMs first.

Acknowledgements

This work was partly supported by an NSF CAREER Award to the second author, by ONR grant N00014-02-1-0932, and by NASA grant NAG 2-1538.

References

[1] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 1989.

[2] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[3] N. Friedman. The Bayesian structural EM algorithm. UAI-98.

[4] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. IJCAI-99.

[5] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. ICML-01.

[6] D. Heckerman, D. Geiger, and D. Chickering. Learning bayesian networks: the combination of knowledge and statistical data. Machine Learning, 1995.

[7] K. Murphy and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In A. Doucet, *et al.*, editors, *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[8] S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. Submitted to IJCAI-03.

A New Perspective of Statistical Modeling by PRISM

Taisuke SATO
Tokyo Institute of Technology / CREST, JST
2-12-1 Ôokayama Meguro-ku Tokyo Japan 152-8552
Neng-Fa Zhou
The City University of New York
2900 Bedford Avenue, Brooklyn, NY 11210-2889

Abstract

PRISM was born in 1997 as a symbolic statistical modeling language to facilitate modeling complex systems governed by rules and probabilities [Sato and Kameya, 1997]. It was the first programming language with EM learning ability and has been shown to be able to cover popular symbolic statistical models such as Bayesian networks, HMMs (hidden Markov models) and PCFGs (probabilistic context free grammars) [Sato and Kameya, 2001]. Last year, we entirely reimplemented PRISM based on a new tabling mechanism of B-Prolog [Zhou and Sato, 2002]. As a result, we can now deal with much larger data sets and more complex models. In this paper, we focus on this recent development and report two modeling examples in statistical natural language processing. One is a declarative PDCG (probabilistic definite clause grammar) program which simulates top-down parsing. The other is a left-corner parsing program which describes a bottom-up parsing that manipulates a stack. The fact that these rather different types of modeling and their EM learning are uniformly possible through PRISM programming shows the versatility of PRISM.[1]

1 Introduction

PRISM[2] was born in 1997 as a symbolic statistical modeling language to facilitate modeling complex systems governed by rules and probabilities [Sato and Kameya, 1997; 2001]. The basic idea is to incorporate a statistical learning mechanism into logic programs for embedded parameters. The result is a unique programming language for symbolic statistical modeling. Actually it was the first programming language with EM learning ability and has been shown to be able to cover popular symbolic statistical models. Theoretically it is an embodiment of Turing machines with learning ability, but the real consequence is that it enables us to build arbitrarily complex symbolic statistical models that may go beyond existing statistical models.

PRISM's power comes from three independent yet interrelated ingredients.

- firm mathematical semantics (*distribution semantics*) [Sato, 1995]
- all solution search using memoizing (OLDT [Tamaki and Sato, 1986] and *linear tabling* [Zhou and Sato, 2002])
- EM learning of parameters embedded in a program by *the graphical EM algorithm* [Kameya and Sato, 2000]

We will not go into the detail of each ingredient, but PRISM has proved to cover most popular statistical models such as HMMs (hidden Markov models) [Rabiner, 1989; Rabiner and Juang, 1993], PCFGs (probabilistic context free grammars) [Wetherell, 1980; Manning and Schütze, 1999] and Bayesian networks [Pearl, 1988; Castillo *et al.*, 1997] with the same time complexity [Sato and Kameya, 2001]. Moreover, we have experimentally confirmed that the learning speed of the graphical EM algorithm [Kameya and Sato, 2000], an EM algorithm for ML (maximum likelihood) estimation employed in PRISM for parameter learning outperforms that of the standard Inside-Outside algorithm for PCFGs by two or three orders of magnitude [Sato *et al.*, 2001].

From the view point of statistical modeling, one of the significant achievements of PRISM is the elimination of the need for deriving new EM algorithms for new applications. When a user constructs a statistical model with hidden variables, all he or she needs is to write a PRISM program using probabilistic built-ins such as `msw/2` predicate representing a parameterized random switch. The remaining work, namely parameter estimation (learning), is taken care of by the graphical EM algorithm quite efficiently thanks to dynamic programming. Furthermore, as long as certain modeling principles are observed, it is mathematically assured that the program correctly performs EM learning (this is *not* self-evident when the model gets complicated). One may say that PRISM is a generic tool for ubiquitous EM learning.

The development of PRISM was gradual because we attempted to fulfill two rather conflicting requirements; exploiting the generality of the semantics and achieving reasonable efficiency for real applications. After all we decided to compromise the generality of semantics and to assume some in-

[1] This paper is partly based on [Sato and Motomura, 2002].

[2] URL= `http://sato-www.cs.titech.ac.jp/prism/index.html`

dependence conditions on programs because while these conditions somewhat restrict the class of acceptable programs, they greatly simplify probability computations thereby making fast EM learning possible.

Our EM learning consists of two phases. In the first preprocessing phase, all solutions are searched for a given goal with respect to a program, yielding a hierarchical graph called an *explanation graph (support graph)*. In the second learning phase, we run the graphical EM algorithm on the explanation graph to train parameters in the program. The graphical EM algorithm is efficient in the sense that it runs in time linear in the size of the explanation graph in each iteration [Sato and Kameya, 2001]. In this learning scheme, compared to the efficiency of the graphical EM algorithm in the learning phase, all solution search in the preprocessing phase could be a bottleneck. A naive search by backtracking would take exponential search time. The key technology to efficiency is memoizing, i.e. to table calls and returns of predicates for later reuse which often reduces exponential time complexity to polynomial time complexity. However, the early versions of PRISM were built on top of SICStus Prolog and it was practically impossible to directly incorporate a full tabling mechanism.

Last year, we replaced the underlying Prolog with B-Prolog and reimplemented PRISM with a full linear tabling mechanism [Zhou and Sato, 2002]. As a result, we can now deal with much larger data sets and more complex models. In this paper, we focus on this recent development and report two modeling examples in statistical natural language processing. One is a declarative PDCG (probabilistic definite clause grammar) program which simulates top-down parsing. The other is a left-corner parsing program which procedurally describes a bottom-up parsing that manipulates a stack. The fact that these rather different types of modeling and their EM learning are uniformly possible through PRISM programming shows the versatility of PRISM.

2 Preliminaries

2.1 A quick review of PRISM

PRISM is a probabilistic extension of Prolog [Sterling and Shapiro, 1986]. A Prolog program is a set of logical formulas called *definite clause*s which take the form H `:-` $B_1, \ldots, B_k$ $(k \geq 0)$. H is an atom called the *head*, and $B_1, \ldots, B_k$ is a conjunction of atoms called the *body*. The clause says if B_1 and $\cdots$ and B_k hold, then H holds (declarative reading). In the context of top-down computation however, it should be read that to achieve goal H, achieve subgoals B_1 and ... and B_k (procedural reading). This twofold reading i.e. bottom-up declarative reading, vs. top-down procedural reading, makes it possible to write declarative but executable programs that encodes both declarative and procedural knowledge in a unified way. When $k = 0$, the clause is called a unit clause. It represents a fact that holds unconditionally. Hence, a collection of ground unit clauses is considered as a relational database.

The surface syntax of PRISM is just Prolog augmented with built-in probabilistic predicates, but the semantics is substantially extended in order to comply with the need of subsuming statistical information in programs. Our semantics guarantees the existence of a unique probability measure, treating every ground atom as a binary random variable.

A PRISM program DB is a set of definite clauses. We write it as $DB = F \cup R$ where F is a set of facts (unit clauses) and R is a set of rules (non-unit clauses) to emphasize the difference of role between facts and rules. One of the unique features of PRISM is that F has a *basic joint probability distribution* P_F.[3] Put it differently, the truth of ground unit clauses $A_1, A_2, \ldots$ in F is probabilistic and their statistical behavior is specified by P_F. Here we consider ground unit clauses as random variables taking on 1 (true) or 0 (false).

What distinguishes our approach from existing approaches to probabilistic semantics is that our semantics admits infinite domains and allows us to use infinitely many random variables (probabilistic ground atoms). Consequently we need not make a distinction between Bayesian networks where a finite number of random variables appear and PCFGs where a countably infinite number of random variables are required. They are just two classes of PRISM programs. Another consequence is that we can implement a variety of EM algorithms as PRISM programs as long as they express, roughly speaking, Turing machines with probabilistic choices.

2.2 Grass_wet example

To put the idea of PRISM across, we show a propositional PRISM program $DB_p = R_p \cup F_p$ in Figure 1.[4] It dose not include any first-order features of PRISM such as logical variables and function symbols.

$$R_p = \left\{ \begin{array}{l} \texttt{g\_wet :- s\_on.} \\ \texttt{g\_wet :- s\_off, w\_rain.} \\ \texttt{g\_dry :- s\_off, w\_clear.} \end{array} \right.$$

$$F_p = \left\{ \begin{array}{l} \texttt{s\_on. s\_off.} \\ \texttt{w\_rain. w\_clear.} \end{array} \right.$$

Figure 1: Wet grass program DB_p

R_p expresses our causal knowledge on the events represented by six propostions: `g_wet` ("grass is wet"), `g_dry` ("grass is dry"), `s_on` ("sprinkler is on"), `s_off` ("sprinkler is off"), `w_rain` ("it is rainy") and `w_clear` ("it is clear"). The first clause says the grass is wet if the sprinkler is on. The second clause says the grass is wet also if the sprinkler is off but the weather is rain. The last clause says the grass is dry if the sprinkler is off and the weather is clear. We assume these rules hold without uncertainty.

In addition to the causal knowledge described above, we know that the states of weather and the sprinkler are probabilistic and are statistically independent. We put this knowledge into the program by setting a probability distribu-

[3] P_F actually is a probability measure over the Herbrand interpretations of F, but for presentation purpose we prefer to use the term "distribution."

[4] This is for the explanatory purpose and not a complete PRISM program. We furthermore need various declarations to run the program.

tion P_{F_p} over random variables s_on, s_off, w_rain and w_clear. When doing so we notice that either s_on or s_off is always true but not both, and this is true of w_rain and w_clear as well. We therefore introduce *parameters* $\theta_s = \text{prob}(\texttt{s\_on} = 1)$ and $\theta_w = \text{prob}(\texttt{w\_rain} = 1)$ and define P_{F_p} as

$$P_{F_p}(\texttt{s\_on} = x_1, \texttt{s\_off} = x_2, \texttt{w\_rain} = x_3, \texttt{w\_clear} = x_4) = \begin{cases} \theta_s^{x_1}(1-\theta_s)^{x_2}\theta_w^{x_3}(1-\theta_w)^{x_4} \\ \qquad \text{if } x_1 + x_2 = 1, x_3 + x_4 = 1 \\ 0 \quad \text{o.w.} \end{cases}$$

Here $x_i \in \{0\,(\text{false}), 1\,(\text{true})\}$ $(1 \leq i \leq 4)$.

Once P_{F_p} is given, the program DB_p defines a joint distribution P_{DB_p} for the six events as follows. Imagine a sample from P_{F_p} and let it be

$$\langle \texttt{s\_on} = 1, \texttt{s\_off} = 0, \texttt{w\_rain} = 0, \texttt{w\_clear} = 1 \rangle.$$

Since the set of true facts ${F'}_p$ is $\{\texttt{s\_on}, \texttt{w\_clear}\}$, it follows that ${F'}_p \cup R_p \vdash \texttt{g\_wet}$ and ${F'}_p \cup R_p \nvdash \texttt{g\_dry}$. In other words, we have $\langle \texttt{g\_wet} = 1, \texttt{g\_dry} = 0 \rangle$. Now we generalize. Let $\langle x_1, x_2, x_3, x_4 \rangle$ be a truth value vector for $\langle \texttt{s\_on}, \texttt{s\_off}, \texttt{w\_rain}, \texttt{w\_clear} \rangle$ sampled from P_{F_p}. Likewise let $\langle y_1, y_2 \rangle$ be a truth value vector for $\langle \texttt{g\_wet}, \texttt{g\_dry} \rangle$. As we saw above, $\langle x_1, x_2, x_3, x_4 \rangle$ determines $\langle y_1, y_2 \rangle$ uniquely, i.e. $\langle y_1, y_2 \rangle$ is a function of $\langle x_1, x_2, x_3, x_4 \rangle$. We denote this function as $\varphi_{DB_p}(\langle x_1, x_2, x_3, x_4 \rangle) = \langle y_1, y_2 \rangle$. Define a joint distribution P_{DB_p} by

$$P_{DB_p}(\texttt{g\_wet} = y_1, \texttt{g\_dry} = y_2, \texttt{s\_on} = x_1, \texttt{s\_off} = x_2, \texttt{w\_rain} = x_3, \texttt{w\_clear} = x_4) \stackrel{\text{def}}{=} \begin{cases} P_{F_p}(x_1, x_2, x_3, x_4) \\ \qquad \text{if } \varphi_{DB_p}(\langle x_1, x_2, x_3, x_4 \rangle) = \langle y_1, y_2 \rangle \\ 0 \quad \text{otherwise} \end{cases}$$

With P_{DB_p} defined now, DB_p becomes a statistical model incorporating logical knowledge. We can calculate whatever probability we need using P_{DB_p}. The parameters θ_s and θ_w are estimated by ML (maximum likelihood) estimation from random observations of s_on, s_off, w_rain and w_clear.

In general, PRISM programs include function symbols, variables and recursion. As a result, the Herbrand domain is infinite and defining P_{DB} is more involved. P_{DB} should be understood as a probability measure over the set of Herbrand interpretations of *DB*, whose cardinality by the way is that of real numbers. Also since parameter learning is ML estimation from incomplete data, we rely on the EM algorithm [Dempster *et al.*, 1977; McLachlan and Krishnan, 1997] for parameter estimation (learning). Mathematical details are explained in [Sato and Kameya, 2001].

3 PDCG

One of the most notable phenomena in natural language processing over the past decade is the adaptation of statistical techniques applied to various corpora [Manning and Schütze, 1999]. In particular probabilistic parsing methods have been developed to tackle the otherwise intractable problem of identifying most plausible parses for a given sentence. Although there are many statistical language models usable for probabilistic parsing, PCFGs have been appreciated as the most basic one due to their simplicity. So we first explain briefly PCFGs [Wetherell, 1980; Manning and Schütze, 1999].

A PCFG (probabilistic context free grammar) is a probabilistic extension of CFG where a CFG rule has a probability. If there are N rules $A \to \alpha_1, \ldots, A \to \alpha_N$ for a non-terminal A, a probability θ_i^A is associated with each rule $A \to \alpha_1$ $(1 \leq i \leq N)$ such that $\sum_{i=1}^N \theta_i^A = 1$. These probabilities are called *parameters* in this paper. Then the probability $p(t)$ of a parse tree t is equal to the product of parameters of rules which are used in the (leftmost) derivation of t. Let T be the set of parse trees for a sentence s. We define the probability $p(s)$ of the sentence s as $p(s) = \sum_{t \in T} p(t)$. When we emphasize $p(s)$ is dependent on the parameters of rules, we write $p(s \mid \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes the set of all parameters.

Below is a simple probabilistic top-down parser written in PRISM a la DCG which is intended to illustrate how easily we can build PCFG like language models (and perform EM learning). The program defines a distribution of provable ground atoms over the form pdcg([s],[w1,...,wn],[]) which corresponds to a sentence [w1,...,wn]. target(pdcg,3) declares what we observe is a predicate pdcg/3.

values/2 declares possible choices for each non-terminal on sentence derivation[5]. For example, values(s,[[ap,vp],[pp,v]]) tells us that the top category s, sentence, has two choices (rules) i.e. s → apvp and s → pp v such that s → ap vp is assigned probability θ_1 and s → pp v probability θ_2 ($\theta_1 + \theta_2 = 1$) respectively. v,n,c,p are terminals and terminal(Wd) says Wd is a terminal whereas first(A,Wd) says Wd is in the first set of the category A. A probabilistic choice is simulated by a built-in predicate msw/2 according to the assigned parameters. For example, when msw(s,RHS) is called in execution mode, s → ap vp will be chosen with probability θ_1. Note that this program is left recursive and would go into an infinite recursion if run by Prolog, but the tabling mechanism of PRISM prevents infinite recursion and realizes CFG parsing with $O(n^3)$ time complexity where n is the sentence length [Sato and Kameya, 2001].

Since the precision of probabilistic parsing by a PCFG is largely determined by the quality of parameters associated with rules in the backbone CFG, their estimation is quite important. Usually it is done by ML estimation from a labeled corpus, i.e. a collection of parse trees). If the corpus is just a collection of sentences (or POS(part of speech) tag sequences), sentences become incomplete data, and it is customarily to appeal to the Inside-Outside algorithm [Baker, 1979; Pereira and Schabes, 1992; Schabes *et al.*, 1993]. In PRISM, the parameters in the above program are estimated by learn/1 built-in predicate. It automatically estimates parameters associated with msw atoms from raw data given

[5] values(s,[v1,...,vk]) declares that a probabilistic switch named s has k choices [v1,...,vk] where s and vi ($1 \leq i \leq k$) are terms. We use this switch s like msw(s,X) in a program when we make a probabilistic choice from [v1,...,vk].

```
target(pdcg,3).
values(s,[[ap,vp],[pp,v]]).
values(vp,[[ap,v],[pp,v]]).
values(np,[[ap,np],[n],
           [np,c,np],[v,n],[vp,n]]).
values(pp,[[n,p],[np,p]]).
values(ap,[[adv],[adv,adv],[adv,np]]).

pdcg([Wd|R],[Wd|L0],L2):-
   terminal(Wd),
   pdcg(R,L0,L2).

pdcg([A|R],[Wd|L0],L2):-
   first(A,Wd),
   ( values(A,[ RHS ])
   ; values(A,[_,_|_]), msw(A,RHS) ),
   pdcg(RHS,[Wd|L0],L1),
   pdcg(R,L1,L2).
pdcg([],L1,L1).
```

Figure 2: A PDCG parser

a list of goals of the form `pdcg([s],[`w_1`,...,`w_n`],[])` by first constructing explanation graphs using tabled search and second running the graphical EM algorithm on them.

The graphical EM algorithm is a generic EM algorithm for PRISM programs and calculates probabilities from explanation graphs, obeying the principle of dynamic programming. It is quite fast. When implemented in C and applied to explanation graphs generated from PCFGs, it runs by far faster than the Inside-Outside algorithm which has been the de facto standard EM algorithm for PCFGs and also runs faster than the Stolcke's EM learning algorithm [Stolcke, 1995], a much more refined EM algorithm based on the Earley parsing model. Experimentally, we observed that when all programs are written in C, the speed ratio[6] of the graphical EM algorithm to the Inside-Outside algorithm is about 1,000:1 and that to the Stolcke's EM learning algorithm is 10:1, depending on grammars [Sato *et al.*, 2001].[7] Unfortunately these speed ratios do not carry over to the graphical EM algorithm implemented in PRISM. This is because the data structure used in PRISM is Prolog terms and hence, we should not expect EM learning by PRISM can match a specialized EM algorithm implemented in C. Nonetheless, just for the record, we report that PRISM installed on a PC (Pentium IV 2.4GHz, 1GB memory, Windows XP) can learn parameters for the ATR grammars (861 CFG rules, 168 nonterminals, 446 terminals) from explanation graphs (95MB in memory) generated from 2,000 sentences of the ATR corpus [Uratani *et al.*, 1994] at a speed of 21 seconds/iteration and the whole learning takes 6,470 seconds (600 seconds for search) in total. We must add however that in this EM learning experiment, we did not use the program in Figure 2 but compiled it to take advantage of Prolog's indexing mechanism for clause invocation. By compilation, the specialized clause for the grammar rule $\mathtt{s} \rightarrow \mathtt{pp\,v}$ looks like

```
pdcg(s,[A|B],C) :-
   first(s,A), msw(s,[pp,v]),
   pdcg(pp,[A|B],D),D=[v|C].
```

We feel that PRISM is becoming competitive with the Inside-Outside algorithm written in C now as far as learning speed is concerned. This is a bit surprising if one considers the fact that PRISM is a much higher level programming language than C. As our implementation still has room for improvement (see our companion paper [Zhou and Sato, 2003] for implementation details), we are expecting to be able to enhance the competitiveness considerably in the near future.

4 Declarative distributions vs. procedural distributions

The language model described by a PCFG is declarative in the sense that the probability of a sentence is directly related to CFG rules, and procedural aspects such as how a parse tree is constructed play no role in calculating the probability of the sentence. This declarative property makes it relatively easy to derive an EM algorithm for PCFGs (and their various extensions like lexicalized PCFGs) and apply it to existing CFG parsers [Stolcke, 1995; Charniak, 1997; Carroll and Rooth, 1998].

When it comes to procedurally defined stochastic CFG parsers, or procedurally defined distributions in general, little work has been done on their EM learning. For example, the GLR (generalized LR(k)) parser [Tomita, 1986] is undoubtedly one of the most sophisticated parsers for natural language processing which performs a sequence of complex stack manipulations while looking up a LR(k) table. Although its probabilistic extension, the PGLR (probabilistic GLR) parser has been proposed in the past [Briscoe and Carroll, 1994; Inui *et al.*, 1997], no EM algorithm is known so far.

This notable contrast can be presumably attributed to the difficultly of formalizing a distribution in terms of operations and their data types employed in the parsing procedure such as stacks, tables, list etc. In the following we present a PRISM program for probabilistic LC (left corner) parsing [Manning, 1997; Roark and Johnson, 1999; Van Uytsel *et al.*, 2001] as an example of the affinity of PRISM programming for procedurally defined distributions. Since PRISM is equipped with a formal semantics and the semantics of a PRISM program is mathematically well-defined, we can be sure of the correctness of EM learning performed by the program no matter how syntactically complicated it is.

5 Probabilistic LC parser

5.1 LC parsing

LC (left corner) parsing is sequential bottom-up parsing for CFG grammars which, like LR(k) parsing, manipulates a

[6] The speed ratio is measured in terms of time required for one iteration.

[7] Theoretically the speed gap is anticipated to widen as the grammars becomes less ambiguous.

stack to reduce subtrees to a larger tree. A program in Figure 3 is a skeletal Prolog LC parser[8]. The top goal is `lc(Ws)` and parsing starts with the subgoal `lc(Ws,[goal(s)])` in the first clause such that `Ws` is a list of words and `s` the starting symbol (sentence). The actual parsing is carried out by `process(Stack0,Stack,L0,L)` in the body of second clause which is tail-recursive.

The parser performs three operations. The shift operation reads a word from the input sentence and pushes it onto a stack which holds nonterminals whose subtrees are completed and subgoals waiting for their corresponding subtrees to complete. The attach operation attaches a completed subtree to the waiting subgoal indicated by `goal/1`. So if a subtree for `B` is completed and if it is waited by a subgoal `goal(B)` at the stack top, `B` is attached to `goal(B)` and the `goal(B)` is removed from the stack. The projection operation treats the completed `B` differently. When `B` is completed, it looks for a CFG rule that has `B` as the left corner category like `A` $\rightarrow$ `B C D` (see `rule(LHS,[B|Rest])` in the third `process/3` clause) and pushes `A`, `goal(D)` and `goal(C)` onto the stack in this order using `predict/3`. Usually top-down pruning is combined with projection and the operation is performed only when `A` is waited for by some subgoal in the stack (this part is not included in the program for simplicity).

```
lc(Ws) :- lc(Ws,[goal(s)]).
lc(L0,Stack0) :-
  process(Stack0,Stack,L0,L),
  lc(L,Stack).

% shift operation
process([goal(C)|Rest],
        [Wd,goal(C)|Rest], [Wd|L], L).

% attach operation
process([B,goal(B)|Stack], Stack, L, L).

% project operation
process([B|Goals], Stack, L, L) :-
  rule(LHS,[B|Rest],
  predict(Rest,[LHS|Goals],Stack).

predict([],L,L).
predict([A|Ls],L2,[goal(A)|NewLs]):-
   predict(Ls,L2,NewLs).
```

Figure 3: A non-probabilistic LC parser

5.2 Probabilistic LC parsing

Probabilistic LC parsing is just a probabilistic version of LC parsing but the point is that it parameterizes CFG rules differently from PCFGs. It assigns probabilities to three operations (shift operation, attach operation and projection operation) in LC parsing. Hence the resulting language distributions form a different class of distributions from those allowed by PCFGs and are expected to be more context sensitive.

Since PRISM programs can be arbitrary Prolog programs, writing a probabilistic LC parser as a PRISM program presents no difficulty to us. Furthermore once we finish writing, it means we have obtained an EM algorithm for LC parsing, provided "due care" is taken to ensure mathematical correctness. That is, the program is written so that it expresses a sequential probabilistic sentence generation process in which every choice is exclusive, independent and made by `msw/2` built-in and once a choice is made, it never leads to failure [Sato and Kameya, 2001].

```
% shift operation
process([goal(A)|Rest],Stack,[Wd|L],L):-
  ( terminal(A),
      A=Wd,Stack=Rest
  ; \+ terminal(A),
      ( values(first(A),[Wd])
      ; values(first(A),[_,_|_]),
          msw(first(A),Wd) ),
      Stack=[Wd,goal(A)|Rest]  ).

% attach or project operation
process([A|Rest],Stack,L,L):-
  \+ A=goal(_),
  Rest=[goal(C)|Stack0],
   ( A==C,
     % goal(A) waits for an A-tree
     ( values(lc(A,A),_),
       % attach and project are possible
         msw(attach(A),Op),
         ( Op==attach,Stack=Stack0
         ; Op==project,
             next_Stack(A,Rest,Stack) )
     ; \+ values(lc(A,A),_),
       % A is forcibly attached
         Stack = Stack0 )
   ; A\==C,
       next_Stack(A,Rest,Stack)  ).

% project operation
next_Stack(A,[goal(C)|Rest2],Stack) :-
    % subtree A is waited for by g(C)
  ( values(lc(C,A),[_,_|_]),
      msw(lc(C,A),rule(LHS,[A|RHS2]))
  ; values(lc(C,A),[rule(LHS,[A|RHS2])]) ),
  predict(RHS2,[LHS,goal(C)|Rest2],Stack).
```

Figure 4: A probabilistic LC parser

[8]This program is taken from [Manning, 1997] with a slight modification.

With this in mind, we replace clauses in Figure 3 for three operations with corresponding ones as in Figure 4 (PRISM declarations for `target/1` and `values/2` are omitted). Since we have to avoid failure in the generation process, program codes are more complicated than the non-probabilistic LC parser.

In a generation process, the shift operation for which the first clause is responsible has two cases depending on whether `A` in `goal(A)` on the stack top is terminal or not. If `A` is a nonterminal and if it has a non-singleton first set, we use `msw(first(A),Wd)` to probabilistically select a word `Wd` to shift[9].

The second clause handles the case where a subtree for nonterminal `A` is completed. There are two cases. The first case is where a subgoal `goal(A)` is waiting on the stack. The other case is where the subtree for `A` has no such waiting subgoal on the stack. The first case is further subdivided into two subcases. In the first subcase, projection is possible[10] as well as attachment. We check this possibility by `values(lc(A,A),_)` and when possible, make a probabilistic choice of the operation. The second subcase is where no such projection is possible and only attach operation is possible.

The project operation is executed in the third clause. When `C` in `goal(C)` on the stack has left-corner relationship with the completed `A` subtree, and if there is more than one rule of the form $\mathtt{A} \rightarrow \mathtt{B}\cdots$, we probabilistically choose one of such rules by `msw(lc(C,A),rule(LHS,[A|RHS2]))`.

The probabilistic LC parser in Figure 4 has no side effects and never fails when used as a sentence generator. It logically describes a sequential decision process where decisions are made by `msw/2` built-in. Consequently, we are sure that the EM learning performed by the program is mathematically correct[11].

We have successfully tested EM learning by the probabilistic LC parser with a small number of data randomly generated from the program itself, but a large scale learning experiment seems difficult because of huge memory requirement. We are developing yet another way to reduce memory requirement using a different formulation of probabilistic LC parsing.

We remark that although there is a formulation of probabilistic LC parsing [Manning, 1997; Roark and Johnson, 1999], the parameter learning there assumes a fully annotated corpus. The only literature we found on the EM learning of LC parsing is [Van Uytsel *et al.*, 2001] in which a specialized EM algorithm for (extended) LC parsing is sketched.

[9] `+X` is Prolog's negation which succeeds if and only if the goal `X` fails. `A==B` succeeds if `A` and `B` are identical Prolog terms whereas `A=B` denotes the unification of `A` and `B`.

[10] In this subcase `A` must have left-corner relationship with itself. In general `A` is said to have left-corner relationship with `A'` if there is a sequence of CFG rules such that $\mathtt{A} \rightarrow \mathtt{B}_1\beta_1, \mathtt{B}_1 \rightarrow \mathtt{B}_2\beta_2, \ldots, \mathtt{B}_n \rightarrow \mathtt{A}'\beta_n$.

[11] To be precise, we need to add a condition "if there is no loss of probability mass to infinite generation process," which is difficult to verify except simple models like PCFGs.

6 Conclusion

Relational learning for uncertainty modeling at first order level is a natural extension of many, if not all, probabilistic approaches and has been developed over the past decade [Breese, 1992; Sato, 1995; Muggleton, 1996; Sato and Kameya, 1997; Koller and Pfeffer, 1997; Cussens, 1999; Friedman *et al.*, 1999; Jaeger, 2001; Sato and Kameya, 2001; Kersting and De Raedt, 2002]. Yet, there seems little work that exploits the full power of the generality of predicate logic combined with statistical learning. Most of work descended from Bayesian networks assumes domains are finite, and dynamic Bayesian networks remain a repetition of the same template. When logic programs are used as an underlying vehicle, range-restrictedness is often imposed which excludes common logic programs such as one for `member` predicate.

PRISM [Sato and Kameya, 1997; 2001] is a general programming language with EM learning ability for modeling symbolic-statistical phenomena. Syntactically it is Prolog augmented with parameterized probabilistic built-ins and accepts *any* programs regardless of whether they are range-restricted or not. Semantically it is the first programming language that can formally define distributions (probability measures) over infinite Herbrand domains. Practically, recent reimplementation of PRISM [Zhou and Sato, 2002] has brought about fast and robust EM learning based on tabled search. The adaptation of B-Prolog's linear tabling mechanism considerably shortens search time and also allows us to use recursive clauses which would otherwise cause infinite recursion, thereby providing us with far more freedom of modeling than previous implementations.

In this paper, we have reported two programming examples in the area of statistical natural language processing that take advantage of this new perspective offered by the latest PRISM. The first one in Figure 2 is a probabilistic DCG program for top-down parsing. It uses difference lists as data structures and accepts left recursive CFG rules. The second one in Figure 4 defines a bottom-up shift-reduce parser, probabilistic LC parser that manipulates a stack. Note that both programs are not range-restricted as logic programs, thus cannot be expressed by those approaches that inhibit non-range-restricted programs. They are not expressible by a fixed size network either because we need an indefinitely many number of random variables that have no upper bound.

Last but not least while we observe that the learning speed of PDCG in PRISM is catching up with the Inside-Outside algorithm implemented in C, it is obvious that we have a lot to do to put PRISM into real use.

References

[Baker, 1979] J. K. Baker. Trainable grammars for speech recognition. In *Proceedings of Spring Conference of the Acoustical Society of America*, pages 547–550, 1979.

[Breese, 1992] J. S. Breese. Construction of belief and decision networks. *Computational Intelligence*, 8(4):624–647, 1992.

[Briscoe and Carroll, 1994] T. Briscoe and J. Carroll. Generalized probabilistic LR parsing of natural language (cor-

pora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59, 1994.

[Carroll and Rooth, 1998] G. Carroll and M. Rooth. Valence induction with a head-lexicalized PCFG. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP 3)*, 1998.

[Castillo *et al.*, 1997] E. Castillo, J. M. Gutierrez, and A. S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer-Verlag, 1997.

[Charniak, 1997] E. Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97)*, 1997.

[Cussens, 1999] J. Cussens. Loglinear models for first-order probabilistic reasoning. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 126–133, 1999.

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Royal Statistical Society*, B39(1):1–38, 1977.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 1300–1309, 1999.

[Inui *et al.*, 1997] K. Inui, V. Sornlertlamvanich, H. Tanaka, and T. Tokunaga. A new probabilistic lr language model for statistical parsing. Technical Report (Dept. of CS) TR97-0004, Tokyo Institute of Technology, 1997.

[Jaeger, 2001] J. Jaeger. Complex probabilistic modeling with recursive relational bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):179–220, 2001.

[Kameya and Sato, 2000] Y. Kameya and T. Sato. Efficient EM learning for parameterized logic programs. In *Proceedings of the 1st Conference on Computational Logic (CL2000)*, volume 1861 of *Lecture Notes in Artificial Intelligence*, pages 269–294. Springer, 2000.

[Kersting and De Raedt, 2002] K. Kersting and L. De Raedt. Basic principles of learning bayesian logic programs. Technical Report Technical Report No. 174, Institute for Computer Science, University of Freiburg, 2002.

[Koller and Pfeffer, 1997] D. Koller and A. Pfeffer. Learning probabilities for noisy first-order rules. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 1316–1321, 1997.

[Manning and Schütze, 1999] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[Manning, 1997] C.D. Manning. Probabilistic parsing using left corner language models. In *Proceedings of the Fifth International Conference on Parsing Technologies (IWPT-97)*, pages 147–158. MIT Press, 1997.

[McLachlan and Krishnan, 1997] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley Interscience, 1997.

[Muggleton, 1996] S. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[Pereira and Schabes, 1992] F. C. N. Pereira and Y. Schabes. Inside-Outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL'92)*, pages 128–135, 1992.

[Rabiner and Juang, 1993] L. R. Rabiner and B. Juang. *Foundations of Speech Recognition*. Prentice-Hall, 1993.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Roark and Johnson, 1999] B. Roark and M. Johnson. Efficient probabilistic top-down and left-corner parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 421–428, 1999.

[Sato and Kameya, 1997] T. Sato and Y. Kameya. PRISM: a language for symbolic-statistical modeling. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 1330–1335, 1997.

[Sato and Kameya, 2001] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[Sato and Motomura, 2002] T. Sato and Y. Motomura. Toward logical-probabilistic modeling of complex systems. In *Proceedings of the International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet*, 2002.

[Sato *et al.*, 2001] T. Sato, S. Abe, Y. Kameya, and K. Shirai. A separate-and-learn approach to EM learning of PCFGs. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium (NLRPS2001)*, pages 255–262, 2001.

[Sato, 1995] T. Sato. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP'95)*, pages 715–729, 1995.

[Schabes *et al.*, 1993] Y. Schabes, M. Roth, and R. Osborne. Parsing the wall street journal with the inside-outside algorithm. In *Proceedings of the 6th European ACL Conference*, pages 341–347, 1993.

[Sterling and Shapiro, 1986] L. Sterling and E. Shapiro. *The Art of Prolog*. The MIT Press, 1986.

[Stolcke, 1995] A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201, 1995.

[Tamaki and Sato, 1986] H. Tamaki and T. Sato. OLD resolution with tabulation. In *Proceedings of the 3rd International Conference on Logic Programming (ICLP'86)*, volume 225 of *Lecture Notes in Computer Science*, pages 84–98. Springer, 1986.

[Tomita, 1986] M. Tomita. *Efficient Parsing for Natural Language*. Kluwer Academic Publishers, 1986.

[Uratani *et al.*, 1994] N. Uratani, T. Takezawa, H. Matsuo, and C. Morita. ATR integrated speech and language database. Technical Report TR-IT-0056, ATR Interpreting Telecommunications Research Laboratories, 1994. In Japanese.

[Van Uytsel *et al.*, 2001] D.H. Van Uytsel, D. Van Compernolle, and P. Wambacq. Maximum-likelihood training of the plcg-based language model. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop 2001 (ASRU'01)*, 2001.

[Wetherell, 1980] C. S. Wetherell. Probabilistic languages: a review and some open questions. *Computing Surveys*, 12(4):361–379, 1980.

[Zhou and Sato, 2002] Neng-Fa Zhou and T. Sato. Toward a High-performance System for Symbolic and Statistical Modeling. Technical Report (Computer Science) TR-200212, City University of New York, 2002.

[Zhou and Sato, 2003] Neng-Fa Zhou and T. Sato. Toward a High-performance System for Symbolic and Statistical Modeling. In *Proceedings of IJCAI-03 workshop on Learning Statistical Models from Relational Data (SRL2003)*, 2003.

Relational Learning: A Web-Page Classification Viewpoint

Seán Slattery
20 Bermondsey Exchange
179-181 Bermondsey Street
London, SE1 3UW
United Kingdom
sean.slattery@cs.cmu.edu

Abstract

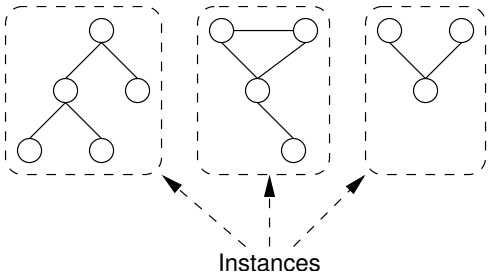
This paper organises some general observations on Relational Learning which arose from research into classifying Web pages. The motivation for this piece is to contribute towards developing a broad overview of the field, so as to understand which aspects of Relational Learning are common to all domains, and which aspects are peculiar to specific domains. Hence the views presented here are necessarily only one piece of the puzzle, and it is hoped that analogous perspectives from other practitioners will improve upon the picture being developed here. With that in mind, let's take a whirlwind tour of Relational Learning as seen through the eyes of someone interested in classifying Web pages.

1 Relational Learning Problems

Relational learning problems, by definition, require data in some relational format. However, there is still some variety in the nature of the classification problems that are considered. Being aware of these variations can be useful when it comes to choosing a particular learning algorithm for the task at hand.

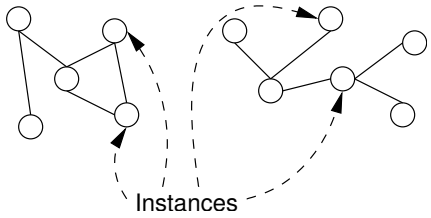
The following general categories of relational learning problem are encountered when considering machine learning on the Web. Do problems from other domains fit into this categorisation? Are there more categories?

1.1 Classifying Graphs



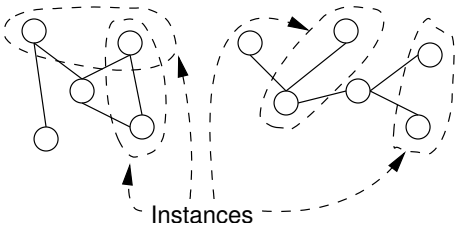
Problems like classifying Web sites fall into this category. Instances are graphs, but usually not connected to each other and the task is to classify the full relational description of an instance.

1.2 Classifying Nodes



Problems like classifying Web pages fall into this category. Instances are nodes, and paths may or may not exist between them. The task is to classify an instance using both the properties of that instance and properties of the "neighbourhood" of that instance. Since nodes can appear in the neighbourhood of more than one instance, thorny independence issues can arise when trying to evaluate whether such shared neighbours are useful predictors or not.

1.3 Classifying Node Tuples



Problems like classifying pairs of Web pages (e.g. is this course being taught by this professor) fall into this category. Instances are tuples of nodes, normally connected in some way. The task is to classify an instance using information about how the nodes relate to each other.

For the moment, I've not broken out edge classification into a separate section. Edge classification on the Web seems to be covered well enough by classifying the pair of web pages linked by the edge as a tuple. Are there compelling examples from other domains that warrant edge classification being considered independently?

2 Relational Learning Features

Being aware of what kinds of regularity your particular relational learning problem may exhibit is important when designing and evaluating algorithms for that problem. This section details some classes of predictive regularity found in Web

page classification problems. Are other forms of regularity found in other relational learning domains, or by algorithms different to the one used in my work (FOIL)?

2.1 Edge Features

My work on Web page classification used a very simple binary predicate representation for hyperlinks between pages. As well as using this predicate to draw neighbouring pages into the hypothesis, the mere existence or non-existence of hyperlinks in certain situations proved to be useful. For example, when learning classify course home pages, FOIL learned this rule

```
course_page(A) :- has_instructor(A), not(has_good(A)),
                  link_to(A,B), not(link_to(B,?)),
                  has_assign(B).
```

This rule uses a binary link_to relation to state that course home pages contain a hyperlink to a page (link_to(A,B)) which contains no hyperlinks to other pages (not(link_to(B,?))). The has_instructor and has_assign predicates are simple keyword predicates, testing for the existence of the words *instructor* and *assign* respectively.

2.2 Node Features

Naturally, properties of individual nodes are also useful features for web page classification. The example in the previous section shows that finding the word *instructor* on a page, and the word *assign* on a page it hyperlinks to, can be an indicator that the page is the home page for a course.

2.3 Aggregate Features

Knowing that a node is linked to "a few" or even "many" nodes which have some feature in common can be useful for classification. For example, index pages of graduate students generally contain many links to Web pages which look like student home pages.

2.4 Related Concept Features

Often for Web page classification tasks, a related concept can be found in the data which helps with the primary task. We've already seen an example with the course rule presented earlier. To classify course home pages, the related concept of "page of assignments" (page containing the work *assign* with no outgoing hyperlinks) exists in the training data and is useful for building a classifier.

3 Relational Learning Algorithms

Many approaches already exist for relational learning and new, more powerful ones are starting to emerge. The work that this paper comes out of used FOIL, a first-order rule learner based on greedy search. These two sections detail some desirables for a relational learning algorithm for Web page classification. How different are the wish lists for other domains?

3.1 Where to look?

When instances are nodes or tuples of nodes, the algorithm has to determine how much and what parts of the neighbourhood contain predictive regularities. One shortcoming of the FOIL approach used is the need for a disparity between positive and negative instances in the *number* of hyperlinks to/from them. In fact, even if such a disparity did exist, the hypothesis language used can only distinguish between "no links" and "at least one link". However, without such a disparity, the algorithm might not have investigated features of hyperlinked pages and not found the interesting regularities it did eventually find. A well implemented relational cliché approach could have neatly solved this problem [Silverstein and Pazzani, 1991].

Not knowing where to look would seem to imply that a relational learning algorithm needs to have some kind of search component, both for building the classification model, and for applying that model to new data.

3.2 What to look for?

The potential for relational learning problems to contain, within them, related concepts is an exciting one. If they do exist for the task at hand, then they are only weakly labeled (e.g. for many course home pages, one of the links from them points to an assignments page) and as such provide an interesting challenge for a learning algorithm.

One interesting idea for a statistical relational learner is to use a latent data approach to label nodes that might belong to a new concept. For example, in a training set for course home pages, we label pages linked to from non-course pages as negative, and treat as latent data the labels on pages linked to from course pages. Perhaps Expectation Maximisation followed by some significance tests on the result could "discover" the concept of assignment pages?

4 Conclusion

This paper presented one view on the world of relational learning based on experience with Web page classification. We looked at the kinds of classification task encountered, the types of regularity that proved useful, and some thoughts on what we'd expect from a good relational learner in this domain (leaving aside practical considerations such as efficiency in space and time).

Analogous views from practitioners in other relational learning domains might serve to give a better intuitive feel for relational learning as a whole, and lead us to better understand what tradeoffs we make with particular relational learning algorithms.

References

[Silverstein and Pazzani, 1991] Glenn Silverstein and Michael J. Pazzani. Relational clichés: Constraining constructive induction during relational learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 203–207, Evanston, IL, 1991.

Statistical Modeling of Graph and Network Data

Padhraic Smyth
Information and Computer Science
University of California, Irvine
CA 92697-3425
smyth@ics.uci.edu

1 Introduction

This brief paper reviews a number of ideas from the statistical and social networks literature that are potentially of interest and relevance to computer scientists working in relational learning. Pointers to references are provided for further reading.

2 Statistical Models for Social Network Data

The statistical literature on social networks typically assumes that we are modeling a set of n entities or "actors" and their binary relationships. The data are often represented in the form of an $n \times n$ adjacency matrix Y where entry $y_{ij} = 1$ (or 0) indicates the presence (or absence) of some form of directed relationship between entities i and j, e.g., "i considers j to be a friend." Undirected graphs, with $y_{ij} = y_{ji}$, are obviously also of interest. More generally y_{ij} can measure the "value of the relation" from entity i to entity j on some suitably defined scale. In addition, each entity can have a set of covariates, denoted $\mathbf{x}_i$, e.g., a vector of demographic measurements, with $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ being the full set of observed covariate data.

There is a long tradition of developing statistical models for such data in the social networks literature (a comprehensive survey of early work in the field is provided by Wasserman and Faust, 1994). Central to these modeling approaches is the treatment of the edge data measurements, y_{ij}, as observations from an underlying distribution for a set of binary random variables defined on each of the ordered $n(n-1)$ pairs i and j. The rationale behind this approach is that the observed $n(n-1)$ relations Y are noisy indicators as to whether a link truly exists or not. The goal of statistical modeling in this context is to infer a parsimonious model for $P(Y|X)$ that requires a relatively small number of parameters to explain the pattern of observed relations (and non-relations), as a function of both local network properties (such as the indegree and outdegree of individual nodes) as well as the covariates X.

Various forms of Markov random fields (MRFs) (Frank and Strauss, 1986) and exponential graph models (sometimes referred to as p^* models in the social networks literature; Wasserman and Pattison, 1996) have been used to model $P(Y|X)$. Much of this work builds on the earlier classic work of Besag in spatial statistics (1974).

These generative modeling frameworks provide all the usual advantages of statistical inference, such as:

- a language for modeling of specific network characteristics, such as reciprocity and transitivity of relations (Wasserman and Faust, 1994);
- modeling techniques for incorporating covariates X, e.g., via suitably-defined logistic regression models;
- inference methods for handling systematic errors in the measurement of links (Butts, 2003);
- hierarchical Bayes and random effect frameworks that allow individual-level variation to be modeled (Hoff, 2003);
- general classes of methods for parameter estimation and model comparison, such as Markov chain Monte Carlo methods (Snijders, 2002);
- incorporation of clusters of nodes in the graph whose statistical properties are equivalent, such as the block models of Wang and Wong (1987);
- methodologies for incorporating specific prior information such as desired functional forms on degree distributions (Snijders, 2003); and
- interpretability of the resulting model (although the validity of such interpretation for certain types of models is debatable: see comments below).

3 The Limitations of Current Statistical Network Models

Unfortunately these models are far from a panacea for all statistical modeling involving network and graph data. Computational issues are a major concern. Parameter estimation in general for Markov random fields is a well-known problem due the intractability of computing the normalization constant in such distributions (which requires, in this context, a sum over all possible graphs with n nodes).

Perhaps even more troubling is the fact that there are essential and fundamental identifiability problems in the estimation of parameters in many of these models—these estimation problems have only become apparent in relatively recent times. To quote Hoff, Raftery, and Handcock (2002):

> "...commonly used models are more global than local in structure and this contributes to model degeneracy and instability problems .... These issues are not resolved by alternative forms of estimation but represent defects in the models themselves...."

Elsewhere, Besag (2002) comments that:

> "A particularly blatant use of MRFs occurs in the analysis of social networks, where the parameters in Markov random graphs are often ascribed substantive interpretations that are meaningless...."

4 Latent Variable Network Models

In this general context, recent work in statistical modeling of network data has begun exploring newer functional forms for $P(Y|X)$ that promise to bypass some of the less desirable features of MRFs. Of particular note, and of potential direct interest to machine learning researchers, is the use of latent (hidden) variable models.

Hoff, Raftery, and Handcock (2002) propose an interesting probabilistic model of this form where the probability of an edge between entities i and j is a function how far apart they are in a k-dimensional latent space. The location vector $\mathbf{z}_i$ for each entity i is estimated from the data, along with parameters that modulate how covariate information (if any) affects the likelihood of an edge between any pair i and j. This leads to models of the form

$$\begin{aligned} & \text{LogOdds } P(y_{ij} = 1 | \mathbf{z}_i, \mathbf{z}_j, \mathbf{x}_i, \mathbf{x}_j, \theta) \\ = \quad & \alpha + f(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\beta}) + d(\mathbf{z}_i, \mathbf{z}_j) \end{aligned}$$

where $f(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\beta})$ is a scalar-valued function dependent on a set of parameters $\boldsymbol{\beta}$ governing pairwise covariate effects (such as the similarity of the characteristics of individual entities) and $d(\mathbf{z}_i, \mathbf{z}_j)$ is a distance in the latent k-dimensional space. A variety of different distance measures (such as Euclidean and absolute) can be used. Hoff and colleagues illustrate how standard maximum likelihood and Bayesian techniques can be applied to this model for parameter estimation. They then apply variations of this model to relatively small social network data sets (the largest data set had 27 entities (nodes)) and achieve relatively interpretable and robust results.

A major limitation of this type of model, however, is the relative lack of scalability. The likelihood is by definition a product over all pairs of nodes, whether an edge was observed or not, leading to an inherently $O(n^2)$ algorithm. While this may be practical for relatively small social networks, the algorithm is not directly scalable to many of the large networks (e.g., with $n = 10^5$) that are often of interest to computer science researchers.

This model is also reminiscent of multi-dimensional scaling (MDS), a well-known technique for "projecting" pairwise similarity data into a multi-dimensional vector space. However, this latent-variable graph model is more powerful than MDS in that the full spectrum of techniques for probabilistic modeling (such as incorporation of covariates) can be brought to bear.

5 Conclusion

It is not yet clear how the classes of statistical models mentioned above are related to other types of models and learning algorithms that have been proposed in the relational learning literature—probabilistic relational models are clearly of particular relevance. In principle, the intersection of statistical modeling and machine learning techniques appears to be a useful area for further exploration. Leveraging the strengths of each approach should produce new classes of models and applications for rich relational domains.

Acknowledgements

The research in this paper was supported by the National Science Foundation under Grant NSF-IIS-0083489.

References

Besag, J. (1974) Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society*, ser B., 36, 192–225.

Butts, C. (2003) Network inference, error, and informant (in)accuracy: a Bayesian approach, *Social Networks*, in press.

Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2002) Latent space approaches to social network analysis, *Journal of the American Statistical Association*, 97, 1090–1098.

Hoff, P. D. (2003) Random effects models for network data, *Proceedings of the National Academy of Sciences*, in press.

Frank, O. and Strauss, D. (1986) Markov graphs, *Journal of the American Statistical Association*, 81, 832–842.

Snijders, T. A. B (2003) Accounting for degree distributions in empirical analysis of network dynamics, *Proceedings of the National Academy of Sciences*, in press.

Snijders, T. A. B (2002) Markov Chain Monte Carlo estimation of exponential random graph models, *Journal of Social Structure*, Vol. 3, No. 2.

Wang, Y. J., and Wong, G. Y. (1987) Stochastic block models for directed graphs, *Journal of the American Statistical Association*, 82, 8–19.

Wasserman, S. and Faust, K. (1994) *Social Network Analysis: Methods and Applications*, Cambridge: Cambridge University Press.

Wasserman, S., and Pattison, P. (1996) Logit models and logistic regression for social networks: I. An introduction to Markov graphs and p^*. *Psychometrika*, 61, 401–425.

Label and Link Prediction in Relational Data

Ben Taskar Pieter Abbeel Ming-Fai Wong Daphne Koller
btaskar, abbeel, mingfai.wong, koller@cs.stanford.edu
Stanford University
Stanford, CA 94305

Abstract

Many real-world domains are *relational* in nature, consisting of a set of entities linked to each other in complex ways. Two important tasks in such data are predicting entity labels and links between entities. We present a flexible framework that builds on (conditional) Markov networks and successfully addresses both tasks by capturing complex dependencies in the data. These models can compactly represent probabilistic patterns over subgraph structures and use them to predict labels and links effectively. We show how to train these models, and how to use approximate probabilistic inference over the learned model for collective classification of multiple related entities and links. We evaluate our framework on several relational datasets, including university webpages and social networks. Our approach achieves significantly better performance than flat classification, which attempts to predict each label and link in isolation.

1 Introduction

The vast majority of work in statistical classification methods has focused on "flat" data – data consisting of identically-structured entities, typically assumed to be independent and identically distributed (IID). However, many real-world data sets are innately relational: hyperlinked webpages, cross-citations in patents and scientific papers, social networks, medical records, and more. Such data consist of entities of many types, where each entity type is characterized by a different set of attributes. Entities are related to each other via different types of links, and the link structure is an important source of information.

Consider a collection of hypertext documents that we want to classify using some set of labels. For example, for a university website, we would like to predict which pages belong to a student, a professor, a course, etc. Most naively, we can use a bag of words model, classifying each webpage solely using the words that appear on the page. However, hypertext has a very rich structure that this approach loses entirely. One document has hyperlinks to others, typically indicating that their topics are related. Each document also has internal structure, such as a partition into sections; hyperlinks that emanate from the same section of the document are even more likely to point to similar documents. When classifying a collection of documents, these are important cues, that can potentially help us achieve better classification accuracy. Therefore, rather than classifying each document separately, we want to provide a form of *collective classification*, where we simultaneously decide on the class labels of all of the entities together, and thereby can explicitly take advantage of the correlations between the labels of related entities.

Another challenge arises from the task of predicting which entities are related to which others and what are the types of these relationships. For example, we might also want to predict not just which page belongs to a professor and which to a student, but also which professor is which student's advisor. In some cases, the existence of a relationship will be predicted by the presence of a hyperlink between the pages, and we will have only to decide whether the link reflects an advisor-advisee relationship. In other cases, we might have to infer the very existence of a link from indirect evidence, such as a large number of co-authored papers. In a very different application, we might want to predict links representing participation of individuals in certain terrorist activities.

One possible approach to this task is to consider the presence and/or type of the link using only attributes of the potentially linked entities and of the link itself. For example, in our university example, we might try to predict and classify the link using the words on the two webpages, and the anchor words on the link (if present). This approach has the advantage that it reduces to a simple classification task and we can apply standard machine learning techniques. However, it completely ignores a rich source of information that is unique to this task — the graph structure of the link graph. For example, a strong predictor of an advisor-advisee link between a professor and a student is the fact that they jointly participate in several projects. In general, the link graph typically reflects common patterns of interactions between the entities in the domain. Taking these patterns into consideration should allow us to provide a much better prediction for links.

A somewhat more sophisticated approach might be to build a link predictor that explicitly takes into consideration other relevant links in the graph. We can implement this either by converting graph features into attributes [10], or by explicitly learning a relational classifier, using techniques such as *inductive logic programming* [8]. Unfortunately, this approach is limited to cases where we are trying to predict a single link at a time, and the other links in the graph are given. In practice, only some or none of the links in the graph

are known, and we are trying to simultaneously predict a large number of links in the graph.

We propose the use of a joint probabilistic model for an entire collection of related entities and links. Following the approach of Lafferty [7], we base our approach on discriminatively trained undirected graphical models, or *Markov networks* [11]. We introduce the framework of *relational Markov network (RMNs)*, which compactly defines a Markov network over a relational data set. The graphical structure of an RMN is based on the relational structure of the domain, and can easily model complex patterns over related entities. For example, we can represent a pattern where two linked documents are likely to have the same topic. We can also capture patterns that involve groups of links: for example, consecutive links in a document tend to refer to documents with the same label. We can also represent "transitive" patterns, where the presence of links A to B and B to C and increases (or decreases) the likelihood of an A to C link. As we demonstrate, RMNs allow tremendous flexibility in representing complex patterns.

Undirected models lend themselves well to discriminative training, where we optimize the conditional likelihood of the labels given the features. Discriminative training, given sufficient data, generally provides significant improvements in classification accuracy over generative training [13]. We provide an effective parameter estimation algorithm for RMNs which uses conjugate gradient combined with approximate probabilistic inference (belief propagation [11]) for estimating the gradient. We also show how to use approximate probabilistic inference over the learned model for collective classification of multiple related entities. We provide experimental results on classification and relation type prediction tasks in web data and a link prediction task in a social network dataset, showing significant gains in accuracy arising from the modeling of relational dependencies.

2 Relational Domains

Consider hypertext as a simple example of a relational domain. A relational domain is defined by a schema, which describes entities, their attributes and relations between them. In our domain, there are two entity types: Page and Hyperlink. If a webpage is represented as a bag of words, Page would have a set of boolean attributes Page.*HasWord*$_k$ indicating whether the word k occurs on the page. It would also have the label attribute Page.*Label*, indicating the topic of the page, which takes on a set of categorical values. The Hyperlink entity type has three attributes: Hyperlink.*Type*, indicating the type of relationship between the two pages, Hyperlink.*From* and Hyperlink.*To*, both of which refer to Page entities.

In general, a *schema* specifies of a set of entity types $\mathcal{E} = \{E_1, \ldots, E_n\}$. Each type E is associated with three sets of attributes: content attributes $E.\mathbf{X}$ (e.g. Page.*HasWord*$_k$), label attributes $E.\mathbf{Y}$ (e.g. Page.*Label*), and reference attributes $E.\mathbf{R}$ (e.g. Hyperlink.*To*). For simplicity, we restrict label and content attributes to take on categorical values. Reference attributes include a special unique key attribute $E.K$ that identifies each entity. Other reference attributes $E.R$ refer to entities of a single type $E' = \mathit{Range}(E.R)$ and take values in $\mathit{Domain}(E'.K)$. Following [4] in addressing link existence prediction, we introduce into our schema object types that correspond to *potential* links between entities. Thus, if we want to reason about all possible links between entities in the model, we can introduce a potential link between every pair of entities in the domain. Each link object has a binary *existence* attribute *Exists*, which is *true* if the link between the associated entities exists and *false* otherwise.

An *instantiation* $\mathcal{I}$ of a schema $\mathcal{E}$ specifies the set of entities $\mathcal{I}(E)$ of each entity type $E \in \mathcal{E}$ and the values of all attributes for all of the entities. For example, an instantiation of the hypertext schema is a collection of webpages, specifying their labels, words they contain and links between them. We will use $\mathcal{I}.\mathbf{X}$, $\mathcal{I}.\mathbf{Y}$ and $\mathcal{I}.\mathbf{R}$ to denote the content, label and reference attributes in the instantiation $\mathcal{I}$; $\mathcal{I}.\mathbf{x}$, $\mathcal{I}.\mathbf{y}$ and $\mathcal{I}.\mathbf{r}$ to denote the values of those attributes. The component $\mathcal{I}.\mathbf{r}$, which we call an *instantiation graph*, specifies the set of entities (nodes) and their reference attributes (edges). A hypertext instantiation graph specifies a set of webpages and links between them, but not their words or labels.

3 Relational Markov Networks

In this section, we present the framework of undirected graphical models, also known as *Markov Networks* [11] or *Markov Random Fields*, and their extension to relational domains.

Markov Networks. Let $\mathbf{V}$ denote a set of discrete random variables and $\mathbf{v}$ an assignment of values to $\mathbf{V}$. A Markov network for $\mathbf{V}$ defines a joint distribution over $\mathbf{V}$. It consists of an undirected dependency graph, and a set of parameters associated with the graph. For a graph G, a *clique* is a set of nodes $\mathbf{V}_c$ in G, not necessarily maximal, such that each $V_i, V_j \in \mathbf{V}_c$ are connected by an edge in G. Note that a single node is also considered a clique.

Definition 1: Let $G = (\mathbf{V}, E)$ be an undirected graph with a set of cliques $C(G)$. Each $c \in C(G)$ is associated with a set of nodes $\mathbf{V}_c$ and a *clique potential* $\phi_c(\mathbf{V}_c)$, which is a non-negative function defined on the joint domain of $\mathbf{V}_c$. Let $\Phi = \{\phi_c(\mathbf{V}_c)\}_{c \in C(G)}$. The Markov net (G, Φ) defines the distribution $P(\mathbf{v}) = \frac{1}{Z} \prod_{c \in C(G)} \phi_c(\mathbf{v}_c)$, where Z is the standard normalizing *partition function*. ∎

Each potential ϕ_c is simply a table of values for each assignment $\mathbf{v}_c$ that defines a "compatibility" between values of variables in the clique. The potential is often represented compactly by a log-linear combination of a small set of indicator functions, or *features*, of the form $f(\mathbf{V}_c) \equiv \delta(\mathbf{V}_c = \mathbf{v}_c)$. In this case, the potential can be written as: $\phi_c(\mathbf{v}_c) = \exp\{\sum_i w_i f_i(\mathbf{v}_c)\} = \exp\{\mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{v}_c)\}$.

For classification, we are interested in constructing discriminative models using *conditional Markov nets* (or conditional random fields [7]), which are simply Markov networks renormalized to model a conditional distribution of some set of target variables $\mathbf{Y}$ given observed variables $\mathbf{X}$: $P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C(G)} \phi_c(\mathbf{y}_c, \mathbf{x}_c)$, where $Z(\mathbf{x})$ is the partition function, now dependent on $\mathbf{x}$.

Relational Markov Networks. A *relational Markov network (RMN)* [12] specifies the cliques and potentials between attributes of related entities at a template level, so a single

model provides a coherent distribution for any collection of instances from the schema. RMNs specify the cliques using *relational clique templates* to identify tuples of variables in the instantiation in a relational query language.

Definition 2: A *relational clique template* $C = (\mathbf{F}, \mathbf{W}, \mathbf{S})$ consists of three components:

- $\mathbf{F} = \{F_i\}$ — a set of object variables, for which $\epsilon(F_i)$ denotes the entity type of F_i.
- $\mathbf{W}(\mathbf{F}.\mathbf{R})$ — a boolean formula using conditions of the form $F_i.R_j = F_k.R_l$.
- $\mathbf{F}.\mathbf{S} \subseteq \mathbf{F}.\mathbf{X} \cup \mathbf{F}.\mathbf{Y}$ — a subset of attributes in $\mathbf{F}$.

For example, if we want to define cliques between the class labels of linked pages, to capture the tendency of pages with the same label tend to link to each other, as in Fig. 1, we might define: $\mathbf{F}$ to be the set *page1*,*page2* and *link* of types Page, Page and Hyperlink, respectively; $\mathbf{W}(\mathbf{F}.\mathbf{R})$ to be $link.From = page1.Key \wedge link.To = page2.Key$; and $\mathbf{F}.\mathbf{S}$ to be $page1.Category$ and $page2.Category$.

A clique template specifies a set of *ground cliques* in an instantiation $\mathcal{I}$:

$$C(\mathcal{I}) \equiv \{c = \mathbf{e}.\mathbf{S} : \mathbf{e} \in \mathcal{I}(\mathbf{F}) \wedge \mathbf{W}(\mathbf{e}.\mathbf{r})\},$$

where $\mathbf{e}$ is a tuple of entities $\{e_i\}$ in which each e_i is of type $\epsilon(F_i)$; $\mathcal{I}(\mathbf{F}) = \mathcal{I}(\epsilon(F_1)) \times \ldots \times \mathcal{I}(\epsilon(F_n))$ denotes the cross-product of entities in the instantiation; the clause $\mathbf{W}(\mathbf{e}.\mathbf{r})$ ensures that the entities are related to each other in specified ways; $\mathbf{e}.\mathbf{S}$ selects the appropriate attributes of the entities.

Definition 3: A *Relational Markov network (RMN)* $\mathcal{M} = (\mathbf{C}, \Phi)$ specifies a set of clique templates $\mathbf{C}$ and corresponding potentials $\Phi = \{\phi_C\}_{C \in \mathbf{C}}$ to define a conditional distribution:

$$P(\mathcal{I}.\mathbf{Y} \mid \mathcal{I}.\mathbf{X}, \mathcal{I}.\mathbf{R}) = \frac{1}{Z} \prod_{C \in \mathbf{C}} \prod_{c \in C(\mathcal{I})} \phi_C(\mathcal{I}.\mathbf{Y}_c, \mathcal{I}.\mathbf{X}_c)$$

where $Z = Z(\mathcal{I}.\mathbf{X}, \mathcal{I}.\mathbf{R})$ is the partition function . ∎

Using the log-linear representation of potentials, $\phi_C(\mathbf{V}_C) = \exp\{\mathbf{w}_C \cdot \mathbf{f}_C(\mathbf{V}_C)\}$, we can write

$$\log P(\mathcal{I}.\mathbf{Y} \mid \mathcal{I}.\mathbf{X}, \mathcal{I}.\mathbf{R}) = \mathbf{w} \cdot \mathbf{f}(\mathcal{I}.\mathbf{Y}, \mathcal{I}.\mathbf{X}, \mathcal{I}.\mathbf{R}) - \log Z$$

where $\mathbf{f}_C(\mathcal{I}.\mathbf{Y}, \mathcal{I}.\mathbf{X}, \mathcal{I}.\mathbf{R}) = \sum_{c \in C(\mathcal{I})} \mathbf{f}_C(\mathcal{I}.\mathbf{Y}_c, \mathcal{I}.\mathbf{X}_c)$ is the sum over all appearances of the template $C(\mathcal{I})$ in the instantiation, and $\mathbf{f}$ is the vector of all $\mathbf{f}_C$.

Given a particular instantiation $\mathcal{I}$ of the schema, the RMN $\mathcal{M}$ produces an *unrolled* Markov network over the attributes of entities in $\mathcal{I}$, in the obvious way. The cliques in the unrolled network are determined by the clique templates C. We have one clique for each $c \in C(\mathcal{I})$, and all of these cliques are associated with the same clique potential ϕ_C.

Probabilistic Models of Graph Structure. The combination of a relational language with a probabilistic graphical model provides a very flexible framework for modeling complex patterns common in relational graphs. First, as observed by Getoor *et al.* [4], there are often correlations between the attributes of entities and the relations in which they participate. For example, in a social network, people with the same hobby are more likely to be friends. More interestingly, we have correlations between the *labels* of entities and the relation type. For example, in a computer science department website, only students can be teaching assistants in a course. We can easily capture such correlations by introducing cliques that involve these attributes. Importantly, these cliques are informative even when attributes are not observed in the test data. For example, if we have evidence indicating an advisor-advisee relationship, our probability that X is a faculty member increases, and thereby our belief that X participates is a teaching assistant in a course Z decreases.

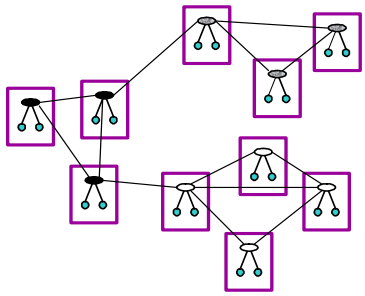


Figure 1: An unrolled Markov net over linked documents. The links follow a common pattern: documents with the same label tend to link to each other more often.

We can also represent much richer patterns over the link graph. Consider, for example, a professor X and two other entities Y and Z. If X's webpage mentions Y and Z in the same context, it is likely that there is correlation between the type of relationship for X-Y and the type for Y-Z. For example, if Y is Professor X's advisee, then probably so is Z. Our framework accommodates these patterns by introducing pairwise cliques between the appropriate relation variables.

Interactions are not limited to pairs of relations. "Transitive" patterns are also common, where the presence of an A-B link and of a B-C link increases (or decreases) the likelihood of an A-C link. For example, students often assist in courses taught by their advisor. Note that this type of interaction cannot be accounted for just using pairwise cliques. By introducing cliques over triples of relations, we can capture such patterns as well. We can incorporate even more complicated patterns, but of course we are limited by the ability of belief propagation to scale up as we introduce larger cliques and tighter loops in the Markov network.

We note that our ability to model these more complex graph patterns relies on our use of an undirected Markov network as our probabilistic model. In contrast, the approach of Getoor *et al.* uses directed graphical models (Bayesian networks and PRMs [6]) to represent a probabilistic model of both relations and attributes. While their approach captures the dependence of link existence on attributes of entities, the constraint that the probabilistic dependency graph be a directed acyclic graph prevents them from representing the more interesting correlations described above.

4 Learning the Models

We focus on the case where the clique templates are given; our task is to estimate the clique potentials, or feature

weights. Thus, assume that we are given a set of clique templates $\mathbf{C}$ which partially specify our (relational) Markov network, and our task is to compute the weights $\mathbf{w}$ for the potentials Φ. In the learning task, we are given some training set D where both the content attributes and the labels are observed. Any particular setting for $\mathbf{w}$ fully specifies a probability distribution $P_{\mathbf{w}}$ over D, so we can use the *likelihood* as our objective function, and attempt to find the weight setting that maximizes the likelihood (ML) of the labels given other attributes. However, to help avoid overfitting, we assume a "shrinkage" prior over the weights (a zero-mean Gaussian), and use maximum a posteriori (MAP) estimation. More precisely, we assume that different parameters are a priori independent and define $p(w_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-w_i^2/2\sigma^2\right\}$.

Both the ML and MAP objective functions are concave and there are many methods available for maximizing them. Our experience is that conjugate gradient and even simple gradient perform very well for logistic regression and relational Markov nets.

Learning Markov Networks. We first consider discriminative MAP training in the flat setting. In this case D is simply a set of IID instances; let d index over all labeled training data D. The discriminative likelihood of the data is $\prod_d P_{\mathbf{w}}(y_d \mid \mathbf{x}_d)$. We introduce the parameter prior, and maximize the log of the resulting MAP objective function:

$$L(\mathbf{w}, D) = \sum_{d \in D} \left(\mathbf{w} \cdot \mathbf{f}(y_d, \mathbf{x}_d) - \log Z(\mathbf{x}_d)\right) - \frac{\mathbf{w} \cdot \mathbf{w}}{2\sigma^2} + C \ .$$

The gradient of the objective function is computed as:

$$\nabla L(\mathbf{w}, D) = \sum_{d \in D} \left(\mathbf{f}(y_d, \mathbf{x}_d) - \boldsymbol{E}_{P_{\mathbf{w}}}[\mathbf{f}(Y_d, \mathbf{x}_d)]\right) - \frac{\mathbf{w}}{\sigma^2} \ .$$

The last term is the shrinking effect of the prior and the other two terms are the difference between the expected feature counts and the empirical feature counts, where the expectation is taken relative to $P_{\mathbf{w}}$:

$$\boldsymbol{E}_{P_{\mathbf{w}}}[\mathbf{f}(Y_d, \mathbf{x}_d)] = \sum_{y'} \mathbf{f}(y'_d, \mathbf{x}_d) P_{\mathbf{w}}(y'_d \mid \mathbf{x}_d) \ .$$

Thus, ignoring the effect of the prior, the gradient is zero when empirical and expected feature counts are equal.[1] The prior term gives the smoothing we expect from the prior: small weights are preferred in order to reduce overfitting. Note that the sum over y' is just over the possible categorizations for one data sample every time.

Learning RMNs. The analysis for the relational setting is very similar. Now, our data set D is actually a single instantiation $\mathcal{I}$, where the same parameters are used multiple times — once for each different entity that uses a feature. A particular choice of parameters $\mathbf{w}$ specifies a particular RMN, which induces a probability distribution $P_{\mathbf{w}}$ over the unrolled Markov network. The product of the likelihood of $\mathcal{I}$ and the parameter prior define our objective function, whose gradient $\nabla L(\mathbf{w}, \mathcal{I})$ again consists of the empirical feature counts minus the expected features counts and a prior term:

$$\mathbf{f}(\mathcal{I}.\mathbf{y}, \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) - \boldsymbol{E}_{\mathbf{w}}[\mathbf{f}(\mathcal{I}.\mathbf{Y}, \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})] - \frac{\mathbf{w}}{\sigma^2}$$

where the expectation $E_{P_{\mathbf{w}}}[\mathbf{f}(\mathcal{I}.\mathbf{Y}, \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})]$ is

$$\sum_{\mathcal{I}.\mathbf{y}'} \mathbf{f}(\mathcal{I}.\mathbf{y}', \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) P_{\mathbf{w}}(\mathcal{I}.\mathbf{y}' \mid \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) \ .$$

This last formula reveals a key difference between the relational and the flat case: the sum over $\mathcal{I}.\mathbf{y}'$ involves the exponential number of assignments to all the label attributes in the instantiation. In the flat case, the probability decomposes as a product of probabilities for individual data instances, so we can compute the expected feature count for each instance separately. In the relational case, these labels are correlated — indeed, this correlation was our main goal in defining this model. Hence, we need to compute the expectation over the joint assignments to all the entities together. Computing these expectations over an exponentially large set is the expensive step in calculating the gradient. It requires that we run inference on the unrolled Markov network.

Inference in Markov Networks. The inference task in our conditional Markov networks is to compute the posterior distribution over the label variables in the instantiation given the content variables. Exact inference algorithms in graphical models can compute this distribution efficiently for specific graph topologies such as sequences, trees and other low treewidth graphs. However, the networks resulting from domains such as hypertext are very large (in our experiments, they contain tens of thousands of nodes) and densely connected. Exact inference is intractable in these cases.

We therefore resort to approximate inference. There is a wide variety of approximation schemes for Markov networks. We chose to use *belief propagation* for its simplicity and relative efficiency and accuracy. Belief Propagation (BP) is a local message passing algorithm introduced by Pearl [11]. It is guaranteed to converge to the correct marginal probabilities for each node only for singly connected Markov networks. Empirical results [9] show that it often converges in general networks, and when it does, the marginals are a good approximation to the correct posteriors. As our results in Section 5 show, this approach works well in our domain.

5 Experiments

We tested our framework on the standard *WebKB* dataset [2] as well as two new real-world datasets which we selected because of interesting relational structure not common to many publicly available datasets. The first new dataset is a WebKB-inspired collection of webpages from several university web sites labeled by rich set of entity and relation categories. The second dataset is a database of university students, including their personal information and lists of their friends.

5.1 WebKB

The data set contains webpages from four different Computer Science departments: Cornell, Texas, Washington and Wisconsin. Each page has a label attribute, representing the type

[1] The solution of maximum likelihood estimation with log-linear models is actually also the solution to the dual problem of maximum entropy estimation with constraints that empirical and expected feature counts must be equal [3].

of webpage which is one of *course, faculty, student, project* or *other*. The data set is problematic in that the category *other* is a grab-bag of pages of many different types. The number of pages classified as *other* is quite large, so that a baseline algorithm that simply always selected *other* as the label would get an average accuracy of 75%. We could restrict attention to just the pages with the four other labels, but in a relational classification setting, the deleted webpages might be useful in terms of their interactions with other webpages. Hence, we compromised by eliminating all *other* pages with fewer than three outlinks, making the number of *other* pages commensurate with the other categories.[2] For each page, we have access to the entire html of the page and the links to other pages. Our goal is to collectively classify webpages into one of these five categories. In all of our experiments, we learn a model from three schools and test the performance of the learned model on the remaining school, thus evaluating the generalization performance of the different models.

Flat Models. The simplest approach we tried predicts the categories based on just the text content on the webpage. The text of the webpage is represented using a set of binary attributes that indicate the presence of different words on the page. We found that stemming and feature selection did not provide much benefit and simply pruned words that appeared in fewer than three documents in each of the three schools in the training data. We also experimented with incorporating meta-data: words appearing in the title of the page, in anchors of links to the page and in the last header before a link to the page [14]. Note that meta-data, although mostly originating from pages linking into the considered page, are easily incorporated as features, i.e. the resulting classification task is still flat feature-based classification. Our first experimental setup compares three well-known text classifiers — Naive Bayes, one-against-others linear support vector machines (Svm), and logistic regression (Logistic) — using words and meta-words. The two discriminative approaches outperform Naive Bayes by an average of nearly 10%. Logistic and Svm give very similar results. The average error over the 4 schools was reduced by around 4% by introducing the meta-data attributes.

Relational Models. Incorporating meta-data gives a significant improvement, but we can take additional advantage of the correlation in labels of related pages by classifying them collectively. We want to capture these correlations in our model and use them for transmitting information between linked pages to provide more accurate classification. We experimented with several relational models. Recall that logistic regression is simply a flat conditional Markov network. All of our relational Markov networks use a logistic regression model locally for each page.

Our first model captures direct correlations between labels of linked pages. These correlations are very common in our data: courses and research projects almost never link to each other; faculty rarely link to each other; students have links to

[2]The category distribution is: course (237), faculty (148), other (332), project (82) and student (542). The numbers of pages/links are: Cornell (280/574), Texas (292/574), Washington (315/728) and Wisconsin (454/1614).

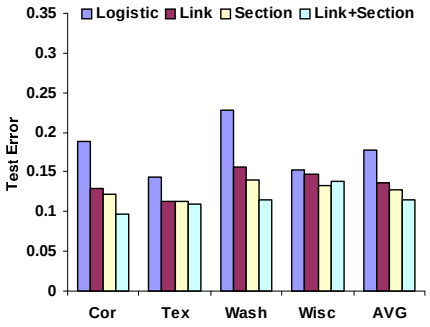


Figure 2: Flat versus collective classification on WebKB: flat logistic regression with meta-data, and three different relational models: Link, Section, and a combined Section+Link.

all categories but mostly courses. The Link model, shown in Fig. 1, captures this correlation through links: in addition to the local bag of words and meta-data attributes, we introduce a relational clique template over the labels of linked pages.

A second relational model uses the insight that a webpage often has internal structure that allows it to be broken up into *sections*. For example, a faculty webpage might have one section that discusses research, with a list of links to all of the projects of the faculty member, a second section might contain links to the courses taught by the faculty member, and a third to his advisees. We can view a section of a webpage as a fine-grained version of Kleinberg's hub [5] (a page that contains a lot of links to pages of particular category). Intuitively, if we have links to two pages in the same section, they are likely to be on similar topics. To take advantage of this trend, we need to enrich our schema with a new relation Section, with attributes *Key*, *Page* (document in which it appears), and *Category*. We also add the attribute *Section* to Hyperlink to refer to the section it appears in. In the RMN, we have two new relational clique templates. The first contains the label of a section and the label of the page it is on:

```
SELECT page.Category, sec.Category
FROM Page page, Section sec
WHERE sec.Page = page.Key
```

The second clique template involves the label of the section containing the link and the label of the target page.

```
SELECT sec.Category, page.Category
FROM Section sec, Hyperlink link, Page page
WHERE link.Sec = sec.Key and link.To = page.Key
```

The original dataset did not contain section labels, so we introduced them using the following simple procedure. We defined a section as a sequence of three or more links that have the same path to the root in the html parse tree. In the training set, a section is labeled with the most frequent category of its links. There is a sixth category *none*, assigned when the two most frequent categories of the links are less than a factor of 2 apart. In the entire data set, the breakdown of labels for the sections we found is: *course* (40), *faculty* (24), *other* (187), *research.project* (11), *student* (71) and *none* (17). Note that these labels are hidden in the test data, so the learning algorithm now also has to learn to predict section labels. Although not our final aim, correct prediction of section

labels is very helpful. Words appearing in the last header before the section are used to better predict the section label by introducing a clique over these words and section labels.

We compared the performance of Link, Section and Section+Link (a combined model which uses both types of cliques) on the task of predicting webpage labels, relative to the baseline of flat logistic regression with meta-data. Our experiments used MAP estimation with a Gaussian prior on the feature weights with standard deviation of 0.3. Fig. 2 compares the average error achieved by the different models on the four schools, training on three and testing on the fourth. We see that incorporating any type of relational information consistently gives significant improvement over the baseline model. The Link model incorporates more relational interactions, but each is a weaker indicator. The Section model ignores links outside of coherent sections, but each of the links it includes is a very strong indicator. In general, we see that the Section models performs slightly better. The joint model is able to combine benefits from both and generally outperforms all of the other models. The only exception is for the task of classifying the Wisconsin data. In this case, the joint Section+Link model contains many links, as well as some large tightly connected loops, so belief propagation did not converge for a subset of nodes. Hence, the results of the inference, which was stopped at a fixed arbitrary number of iterations, were highly variable and resulted in lower accuracy.

5.2 Extended WebKB

We collected and manually labeled a new relational dataset inspired by WebKB [2]. Our dataset consists of Computer Science department webpages from 3 schools: Stanford, Berkeley, and MIT.

A total of 2954 of pages are labeled into one of eight categories: faculty, student, research scientist, staff, research group, research project, course and organization (organization refers to any large entity that is not a research group). *Owned pages*, which are owned by an entity but are not the main page for that entity, were manually assigned to that entity. The average distribution of classes across schools is: organization (9%), student (40%), research group (8%), faculty (11%), course (16%), research project (7%), research scientist (5%), and staff (3%).

We established a set of candidate links between entities based on evidence of a relation between them. One type of evidence for a relation is a hyperlink from an entity page or one of its owned pages to the page of another entity. A second type of evidence is a *virtual link*: We assigned a number of aliases to each page using the page title, the anchor text of incoming links, and email addresses of the entity involved. Mentioning an alias of a page on another page constitutes a virtual link. The resulting set of 7161 candidate links were labeled as corresponding to one of five relation types — Advisor (faculty, student), Member (research group/project, student/faculty/research scientist), Teach (faculty/research scientist/staff, course), TA (student, course), Part-Of (research group, research proj) — or "none", denoting that the link does not correspond to any of these relations.

The observed attributes for each page are the words on the page itself and the "meta-words" on the page — the words

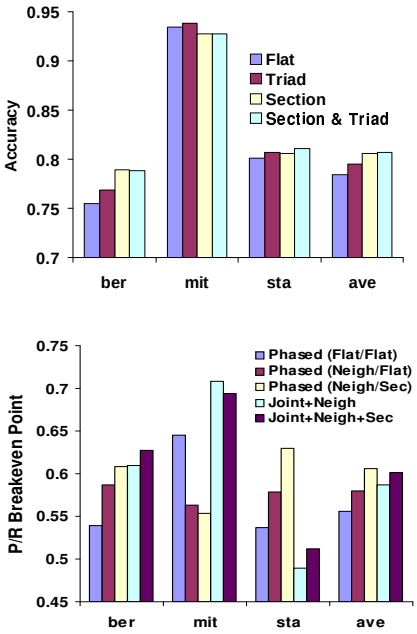


Figure 3: Top: Relation prediction with entity labels given. Bottom: Relation prediction without entity labels.

in the title, section headings, anchors to the page from other pages. For links, the observed attributes are the anchor text, text just before the link (hyperlink or virtual link), and the heading of the section in which the link appears.

Our task is to predict the relation type, if any, for all the candidate links. We tried two settings for our experiments: with page categories observed (in the test data) and page categories unobserved. For all our experiments, we trained on two schools and tested on the remaining school.

Observed Entity Labels. We first present results for the setting with observed page categories. Given the page labels, we can rule out many impossible relations; the resulting label breakdown among the candidate links is: none (38%), member (34%), part-of (4%), advisor (11%), teach (9%), TA (5%).

We tried several models. Link-Flat is our baseline model: it assumes relations are independent of each other, so we use multinomial logistic regression as our classifier. The features used by this model are the labels of the two linked pages and the words on the links going from one page and its owned pages to the other page. The number of features is ≈ 1000.

The relational models try to improve upon the baseline model by modeling the interactions between relations and predicting relations jointly. The Section model introduces cliques over relations whose links appear consecutively in a section on a page. This model tries to capture the effect that similarly related entities (e.g., advisees, members of projects) are often listed together on a webpage. The Triad model tries to capture higher order patterns involving triples of related entities, as discussed in Section 3. Specifically, we introduce cliques over sets of three candidate links that form a triangle in the link graph. The Section + Triad model includes the cliques of the two models above.

As shown in Fig. 3 (top), both the Section and Triad mod-

els outperform the flat model, and the combined model has an average accuracy gain of 2.26%, or 10.5% relative reduction in error. As only a subset of all candidate links is affected by the section or triad cliques, we also computed the average accuracy gain on just the links involved in these richer cliques. We obtained an improvement of 3.78% on the 50% of links affected by Section.

As an example of the interesting inferences made by the models, we found a student-professor pair that was misclassified by Flat model as none (there is only a single hyperlink from the student's page to the advisor's) but correctly identified by both the Section and Triad models. The Section model utilizes a paragraph on the student's webpage describing his research, with a section of links to his research groups and the link to his advisor. Examining the parameters of the Section model clique, we found that the model learned that it is likely for people to mention their research groups and advisors in the same section. By capturing this trend, the Section model is able to increase the confidence of the student-advisor relation. The Triad model corrects the same misclassification in a different way. Using the same example, the Triad model makes use of the information that both the student and the teacher belongs to the same research group, and the student TAed a class taught by his advisor. It is important to note that none of the other relations are observed in the test data, but rather the model bootstraps its inferences.

Unobserved Entity Labels. When the labels of pages are not known during relations prediction, we cannot rule out possible relations for candidate links based on the labels of participating entities. Thus, we have many more candidate links that do not correspond to any of our relation types (e.g., links between an organization and a student). This makes the existence of relations a very low probability event, with the following breakdown among the potential relations: none (71%), member (16%), part-of (2%), advisor (5%), teach (4%), TA (2%). In addition, when we construct a Markov network in which page labels are not observed, the network is much larger and denser, making the (approximate) inference task much harder. Thus, in addition to models that try to predict page entity and relation labels simultaneously, we also tried a two-phase approach, where we first predict page categories, and then use the predicted labels as features for the model that predicts relations.

For predicting page categories, we compared two models. Entity-Flat model is multinomial logistic regression that uses words and "meta-words" from the page and its owned pages in separate "bags" of words. The number of features is roughly $10,000$. The Neighbors model is a relational model that also exploits regularities in web site organization: pages with similar urls often belong to the same category or tightly linked categories (research group/project, professor/course). For each page, two pages with urls closest in edit distance are selected as "neighbors", and we introduced pairwise cliques between "neighboring" pages. The Neighbors model outperforms the Flat model across all schools by an average of 4.9% accuracy gain.

Given the page categories, we can now apply the different models for link classification. Thus, the Phased (Flat/Flat) model uses the Entity-Flat model to classify the page labels, and then the Link-Flat model to classify the candidate links using the resulting entity labels. The Phased (Neighbors/Flat) model uses the Neighbors model to classify the entity labels, and then the Link-Flat model to classify the links. The Phased (Neighbors/Section) model uses the Neighbors to classify the entity labels and then the Section model to classify the links.

We also tried two models that predict page and relation labels simultaneously. The Joint + Neighbors model is simply the union of the Neighbors model for page categories and the Flat model for relation labels given the page categories. The Joint + Neighbors + Section model additionally introduces the cliques that appeared in the Section model between links that appear consecutively in a section on a page. We train the joint models to predict page and link labels simultaneously.

Since the proportion of relation "none" is so overwhelming, we use the following decision rule to classify relations: If the probability of "none" is less than a given threshold, predict the most likely label (other than none), otherwise predict the most likely label (including none). We report precision recall breakeven point using this rule with the threshold set to a value where precision of actual relations (of all types except none) equals recall on the test data. Fig. 3 (bottom) compares the resulting breakeven points achieved by the different models on the three schools. Relational models, both phased and joint, did better than flat models on the average. However, performance varies from school to school and for both joint and phased models, performance on one of the schools is worse than that of the flat model.

5.3 Social Network Data

The second dataset we used has been collected by a portal website at Stanford university that hosts an online community for students [1]. Among other services, it allows students to enter information about themselves, create lists of their friends and browse the social network. Personal information includes residence, gender, major and year, as well as favorite sports, music, books, social activities, etc. We focused on the task of predicting the "friendship" links between students from their personal information and a subset of their links. We selected students living in sixteen different residences or dorms and restricted the data to the friendship links only within each residence. Each residence has about 15–25 students and an average student lists about 25% of housemates as friends.

We used an eight-fold train-test split, where we trained on fourteen residences and tested on two. Note that the students in the training and test set are disjoint and no links between them exist. Predicting links between two students from just personal information alone is a very difficult task, so we tried a more realistic setting, where some proportion of the links is observed in the test data, and can be used as evidence for predicting the remaining links. We used the following proportions of observed links in the test data: 10%, 25%, and 50%. The observed links were selected at random, and the results we report are averaged over five folds of these random selection trials.

Using just the observed portion of links, we constructed the following features: for each student, the proportion of stu-

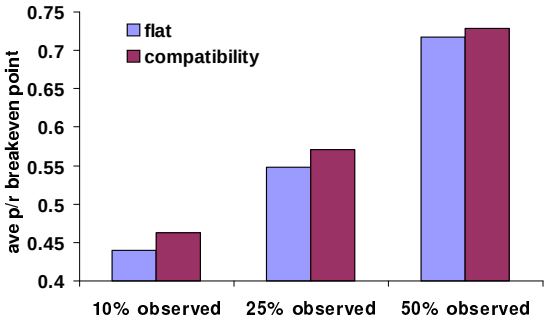


Figure 4: Average precision/recall breakeven point for 10%, 25%, 50% observed links.

dents in the residence that list him/her and the proportion of students he/she lists; for each pair of students, the proportion of other students they have as common friends. The values of the proportions were discretized into four bins. These features capture some of the relational structure and dependencies between links: Students who list (or are listed by) many friends in the observed portion of the links tend to have links in the unobserved portion as well. More importantly, having friends in common increases the likelihood of a link between a pair of students.

The Flat model uses logistic regression with the above features as well as personal information about each user. In addition to characteristics of the two people, we also introduced a feature for each match of a characteristic, e.g. both people are computer science majors or both are freshmen.

The Compatibility model (which resembles our Section model above) introduces cliques between each pair of links emanating from each person. Similarly to the Flat model, these cliques include a feature for each match of the characteristics of the two potential friends. This model captures the tendency of a person to have friends who share many characteristics (even though the person might not possess them). We also tried models with triad cliques, but the belief propagation often failed to converge, producing erratic results.

Fig. 4 compares the average precision/recall breakpoint achieved by the different models at the three different settings of observed links. The Compatibility model outperforms the flat with p-values 0.0036, 0.00064 and 0.054 respectively, according to a paired t-test.

6 Discussion and Conclusions

We address the novel task of collective label and link classification, where we are simultaneously trying to predict and classify an entire set of labels and links in a link graph. Our approach provides a coherent probabilistic foundation for this task, by defining a joint probabilistic model over objects and links. Our framework allows us to represent a very rich set of relational patterns in the probabilistic model, and use them in prediction. The resulting models significantly improve the classification accuracy over flat models.

Our results in this paper are only a first step toward understanding the power of relational classification, and many extensions are possible. On the representational side, we can gain significant power from introducing hidden variables (that are not observed even in the training data). In a different extension, one of the problems limiting the applicability of approach is the reliance on belief propagation, which often does not converge in more complex problems. We believe that this issue can be addressed if we consider a tighter integration of learning and inference.

Our results use a set of relational patterns that we have discovered to be useful in the domains that we have considered. However, many other rich and interesting patterns are possible. Thus, in the relational setting, the issue of feature construction is critical. It is therefore important to explore the problem of automatic feature induction, as in [3].

Finally, we believe that this framework can provide a principled approach for addressing a wide range of applications, including predicting communities of people and hierarchical structure of people and organizations, based on both the local attributes and the patterns of static and dynamic interaction.

References

[1] L. Adamic, O. Buyukkokten, and E. Adar. A social network caught in the web. http://www.hpl.hp.com/shl/papers/social/, 2002.

[2] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proc AAAI98*, pages 509–516, 1998.

[3] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[4] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Probabilistic models of relational structure. In *Proc. ICML01*, Williamstown, MA., 2001.

[5] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. of the ACM*, 46(5):604–632, 1999.

[6] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proc. AAAI98*, pages 580–587, Madison, Wisc., 1998.

[7] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML01*, 2001.

[8] Nada Lavrač and Saso Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, New York, 1994.

[9] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proc. UAI99*, pages 467–475, 1999.

[10] J. Neville and D. Jensen. Iterative classification in relational data. In *Proc. AAAI00 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.

[11] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.

[12] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proc. UAI02*, Edmonton, Canada, 2002.

[13] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, New York, 1995.

[14] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *J. of Intelligent Information Systems*, 18(2), 2002.

Toward a High-performance System for Symbolic and Statistical Modeling

Neng-Fa Zhou
Department of Computer Science
Brooklyn College & Graduate Center
The City University of New York
zhou@sci.brooklyn.cuny.edu

Taisuke Sato
Department of Computer Science,
Tokyo Institute of Technology, CREST JST
sato@mi.cs.titech.ac.jp

Koiti Hasida
Cyber Assist Research Center
AIST Tokyo Waterfront
hasida.k@aist.go.jp

Abstract

We present in this paper a state-of-the-art implementation of PRISM, a language based on Prolog that supports statistical modeling and learning. We start with an interpreter of the language that incorporates a naive learning algorithm, and then turn to improve the interpreter. One of the improvements is to refine the learning algorithm such that it works on explanation graphs rather than flat explanations. Tabling is used to construct explanation graphs so that variant subgoals do not need to be considered redundantly. Another technique is compilation. PRISM programs are compiled into a form that facilitates searching for all solutions. The implemented system is, to our knowledge, the first of its kind that can support real-world applications. The implemented system, which will be available from *http://sato-www.cs.titech.ac.jp/prism/index.html*, is being applied to several problem domains ranging from statistical language processing, decision support, to game analysis.

1 Introduction

PRISM (PRogramming In Statistical Modeling) [Sato, 1995; Sato and Kameya, 2001] is a new language that integrates probability theory and Prolog, and is suitable for the description of computations in which randomness or uncertainty is involved. PRISM provides built-ins for describing experiments [1]. A PRISM program can be executed in three different modes, namely *sample execution*, *probability calculation*, and *learning*. In sample execution mode, a goal may give different results depending on the outcomes of the experiments. For example, it is possible for a goal to succeed if a coin shows the head after being tossed and to fail if the coin shows the tail. The probability calculation mode gives the probability of a goal to succeed. In the learning mode, the system estimates the probabilities of the outcomes of the experiments from given observed data. The PRISM system adopts the EM (Expectation and Maximization) algorithm [Dempster *et al.*, 1976] in probability estimation.

PRISM, as a symbolic statistical modeling language, subsumes several specific statistical tools such as HMM (Hidden Markov Models) [Rabiner, 1989], PCFG (Probabilistic Context Free Grammars) [Wetherell, 1980] and descrete Bayesian networks [Castillo *et al.*, 1997; Pearl, 1987]. Compared with numeric models where mathematical formulas are used, PRISM offers incomparable flexibility by allowing the use of arbitrary logic programs to describe probability distributions. PRISM can be used in many areas such as language processing, decision making, bio-informatics, and game theory where randomness or uncertainty is essential.

This project aims at implementing an efficient system for PRISM in B-Prolog. For most applications, learning is time-consuming especially when the amount of observed data is large. The EM learning algorithm estimates the probabilities of outcomes through two phases: the first phase searches for all explanations for the observed facts, and the second phase estimates the probabilities. The first phase is the neck of the learning algorithm. We have made several efforts to speed-up this phase. One is to tabulate partial explanations for subgoals such that explanations for variant subgoals are searched only once. With tabling, this phase gives an *explanation graph* that facilitates the estimation of probabilities. The tabling mecha-

[1] An experiment is defined by a sample space and a probability distribution for the outcomes in the sample space. For example, tossing a coin is an experiment where the sample space is {`head`, `tail`} and the probability distribution is uniform (this means that the events `head` and `tail` have the same likelihood to occur) if the coin is fair.

nism of B-Prolog is improved such that copy of data between the heap and the tabling area is reduced significantly. This improved version demonstrates a big speed-up when complex goals with structured data need to be tabulated. Another technique used in the system is compilation. PRISM programs are compiled into a form that facilitates searching for all solutions.

The main part of this paper is devoted to the implementation techniques. To make the paper self-contained, we start with an interpreter of PRISM in the next section. The description of the operational semantics is informal and is based on examples. The reader is referred to [Sato and Kameya, 2001] for a formal description of the semantics and the EM learning algorithm adopted in PRISM.

2 PRISM: The Language and its Implementation

PRISM is an extension of Prolog that provides built-ins for statistical modeling and learning.

2.1 Built-ins

The built-in `msw(I,V)` describes a trial of an *experiment*, where `I` is the identifier of an experiment, and `V` is the outcome of the trial[2]. The identifier `I` can be any complex term, but `I` must be ground when the trial is conducted. In the sample-execution mode, the built-in `msw(I,V)` succeeds if the trial of the experiment `I` gives the outcome `V`. If `V` is a variable, then the built-in always succeeds, binding `V` to the outcome of the experiment.

For each experiment, the user must specify the *sample space* by defining the predicate `values(I,Space)`, where `I` is the identifier and `Space` is a list of possible outcomes of the experiment. A *probability distribution* of an experiment tells the probabilities of the outcomes in the sample space. The sum of the probabilities of the outcomes in any experiment must be 1.0. Probability distributions are either given by the programmer or obtained through learning from given *sample data*. The predicate `set_sw(I,Probs)` sets the probabilities of the outcomes in the experiment `I`, where `Probs` is a list of probabilities (floating-point numbers). The length of `Probs` must be the same as the number of outcomes in the sample space and the sum of the probabilities must be equal to 1.0.

The following shows an illustrative example:

```
direction(D):-
   msw(coin,Face),
   (Face==head->D=left;D=right).

values(coin,[head,tail]).
```

[2]The name *msw* is an abbreviation for *multi-outcome switch*. In the version presented in [Sato and Kameya, 2001], the built-in takes another argument called *trial number*. The same trial of the same experiment must give the same outcome. In the new version, all trials are considered independent by default. If the outcome of a trial needs to be reused, the programmer must have it passed as an argument or have it saved in the global database.

The predicate `direction(D)` determines the direction to go by tossing a coin; D is bound to `left` if the head is shown, and to `right` if the tail is shown. To set uniform distribution, we use `set_sw(coin,[0.5,0.5])` to set the probabilities to the two outcomes. Notice that the following gives a different definition of `direction`:

```
direction(left):-
   msw(coin,head).
direction(right):-
   msw(coin,tail).
```

While for the original definition, the query `direction(D)` always succeeds, binding `D` to either `left` or `right`. The same query may fail for the new definition since `msw(coin,head)` and `msw(coin,tail)` are two separate trials. If the first trial gives `tail` and the second trial gives `head`, then the query `direction(D)` fails.

In addition to `msw/2`, PRISM provides several other built-ins, including `prob(Goal,Prob)` for computing the probability of a goal, `sample(Goal)` for sample executing a goal, and `learn(Facts)` for estimating the probabilities of the switches in the program from the observed facts. These built-ins will be explained in the subsequent subsections.

A predicate is said to be *probabilistic* if it is defined in terms of `msw` or predicates that are probabilistic. Predicates that do not use (either directly or indirectly) `msw` in its definition are said to be *non-probabilistic*. This terminology is extended naturally to goals. A goal is said to be *probabilistic* if its predicate is probabilistic.

2.2 Sample execution

The subgoal `sample(Goal)` starts executing the program with respect to `Goal` in the sample execution mode. If `Goal` is the built-in `msw(I,V)`, then `sample(Goal)` succeeds if the trial of the experiment `I` gives the outcome `V`. The outcome of an experiment is chosen randomly, but the probability distribution is respected such that those outcomes that have the highest probabilities have the most chances to be chosen. Trials of experiments are independent regardless of whether or not the experiments are the same.

If `Goal` is non-probabilistic, then `sample(Goal)` behaves in the same way as `call(Goal)`. Otherwise, if `Goal` is probabilistic, then a clause `H:-Body` is selected from its predicate such that `H` unifies `Goal`, and `sample(Goal)` is reduced to `sample(Body)`.

The following shows a simplified version of the interpreter for sample execution:

```
sample((A,B)):-!,
   sample(A),
   sample(B).
sample(msw(I,V)):-!,
   R is random(0.0,1.0),
   % R is a random number in the range of 0.0..1.0
   prob_distribution(I,Values,Probs),
   % probability distribution assigned to the experiment
   choose_outcome(R,Values,Probs,V).
sample(Goal):-prob_predicate(Goal),!,
   clause(Goal,Body),
   sample(Body).
```

```
sample(Goal):-  % non-probablistic
   call(Goal).

choose_outcome(R,Values,Probs,V):-
   choose_outcome(R,0.0,Values,Probs,V).
choose_outcome(R,Sum,[V|Values],[P|Probs],V):-
   Sum1 is Sum+P,
   R=<Sum1,!.
choose_outcome(R,Sum,[_|Values],[P|Probs],V):-
   Sum1 is Sum+P,
   choose_outcome(R,Sum1,Values,Probs,V).
```

For an experiment whose sample space is `[V1,V2,...,Vn]` and whose probability distribution is `[P1,P2,...,Pn]`, the call

```
choose_outcome(R,[V1,V2,...,Vn],[P1,P2,...,Pn],V)
```

selects the outcome `Vk` such that $\sum_{i=1}^{k} Pi \geq R$ and $\sum_{i=1}^{k-1} Pi < R$.

The real interpreter handles other constructs including negation, disjunction, if-then-else, and the cut operator in addition to conjunction.

2.3 Calculating the probabilities of goals

In statistical modeling, it is often necessary to calculate the probability of events. In PRISM, the built-in `prob(Goal,Prob)` calculates the probability `Prob` with which `Goal` becomes true. It is assumed that all probabilistic ground atoms in the Herbrand base of a program are probabilistically *independent* and *exclusive*. With these assumptions, the probability of the conjunction `(A,B)` is computed as the product of the probabilities of `A` and `B` (*independent*), and the probability of the disjunction `(A;B)` is computed as the sum of the probabilities of `A` and `B` (*exclusive*). For a switch `msw(I,V)`, the probability is 1.0 if `V` is a variable, and the probability assigned to the outcome `V` if `V` an element is the sample space.

For example, recall the illustrative example `direct`. Assume the distribution of the `coin` experiment is uniform. The probability of `direction(left)` is `0.5` since the probability of `msw(coin,head)` is `0.5`. The probability of `direction(D)` is 1.0 sine the sum of the probabilities of `msw(coin,head)` and `msw(coin,tail)` is 1.0.

The programmer must bear the above assumptions in mind when writing programs. Programs that violates this assumption will give wrong results. For example, the conjunction `(A,A)`, which makes sense logically, is not allowed probabilistically since the conjuncts are not independent. Likewise the disjunction `(A;A)` is not allowed. If the disjuncts are not independent, the probability of a goal may exceed 1.0.

One question arises: if events are assumed to be independent, then how to represent conditional events in PRISM? Let B and C be two experiments. Assume C has the possible outcomes $\{c_1, ..., c_n\}$. The conditional event $(B|C)$ can be represented by using n switches: $msw(b(c_i), V_i)$ (i=1,...,n). Consider, for example, the following problem taken from [Stirzaker, 1994], which is a typical example of Bayesian reasoning.

> You have a blood test for some rare disease which occurs by chance in 1 in every 100,000 people. The test is fairly reliable; if you have the disease it will correctly say so with probability 0.95; if you do not have the disease, the test will wrongly say you do with probability 0.005. If the test says you do have the disease, what is the probability that this is a correct diagnosis?

Let D be the event that you have the disease, D' the event that you do not have the disease, and T the event that the test says you do. Then the probability $P(D|T)$ is calculated as follows based on the Bayes' Theorem:

$$\begin{aligned} P(D|T) &= \frac{P(T|D)P(D)}{P(T)} \\ &= \frac{P(T|D)P(D)}{P(T|D)P(D)+P(T|D')P(D')} \\ &= \frac{0.95\times 0.00001}{0.95\times 0.00001+0.005\times 0.99999} \\ &= 0.1896 \end{aligned}$$

The Bayesian network for this problem consists of two nodes, called `disease` and `test`. The outcomes of both nodes are {`yes, no`}. The node `test` is dependent on the node `disease`. The following clause represents the network:

```
disease_test(D,T):-
   msw(disease,D),
   msw(test(D),T).
```

The sample spaces of all the experiments are `[yes,no]`. The switch `msw(disease,yes)` says that you have the disease, and the switch `msw(disease,no)` says no. The switch `msw(test(D),T)`, which depends on the outcome of the node `disease`, says that the diagnostic result is `T` if the outcome of `disease` is `D`. For the problem, the given probabilities are set as follows:

```
set_sw(disease,[0.00001,0.99999]),
   % P(D)=0.00001
set_sw(test(yes),[0.95,0.05]),
   % P(T|D)=0.95
set_sw(test(no),[0.005,0.995])
   % P(T|D')=0.005
```

If the test says you do have the disease, then the probability that this is a correct diagnosis is calculated by the query:

```
prob(disease_test(yes,yes),P1),
prob(disease_test(_,yes),P2),
P is P1/P2.
```

The goal `prob(disease_test(yes,yes),P1)` gives the probability of the event that you have the disease and is also diagnosed so, and the goal `prob(disease_test(_,yes),P2)` gives the probability of the event that you are diagnosed of the disease regardless whether or not you have the disease. The query gives the same result 0.1896 as the one obtained by using Bayes' Theorem directly.

Since new switches can be created when needed, it is possible to represent in PRISM any Bayesian networks and perform Bayesian reasoning on them.

2.4 Learning

The built-in `learn(Facts)` takes `Facts`, a list of observed facts, and estimates the probabilities of the switches that explain `Facts`. While `sample(Goal)` and `prob(Goal,Prob)` are deductive, using the current distributions of switches to deduct `Goal`, `learn(Facts)` is abductive, which finds the explanations for `Facts` and use the explanations to estimate the distributions of the switches.

PRISM adopts the EM learning algorithm to learn distributions. It first finds all the explanations for the observed facts. Then it repeatedly estimates and maximizes the likelihood of the observed facts until the estimation is stable.

An *explanation* for an observed fact is a set of switches that occur in a path of the execution of the fact. The following is an interpreter that searches for explanations for a goal:

```
expls(G,Exs):- %Exs is a list of explanations for G
   findall(Ex,expl(G,Ex,[]),Exs).
expl((G1,G2),Ex,ExR):-!,
   expl(G1,Ex,Ex1),
   expl(G2,Ex1,ExR).
expl(msw(I,V),[mse(I,V)|ExR],ExR):-!,
   values(I,Values), % sample space is Values
   member(V,Values).
expl(G,Ex,ExR):-
   prob_predicate(G),!, %G is a probabilistic
   clause(G,B),
   expl(B,Ex,ExR).
expl(G,Ex,Ex):-
   call(G).
```

Recall our illustrative example `direction`. For the fact `direction(left)`, the interpreter finds `[msw(coin,head)]`, and for the fact `direction(right)` it finds `[msw(coin,tail)]` as the explanations. In general, there may exist multiple execution paths for an observed fact and each execution path may contain multiple switches.

After all the explanations are found, the EM algorithm turns to estimate the probabilities of the switches in the explanations. Let I be the set of switches, and V_i be the sample space of switch i. For each switch $msw(i,v)$, $\theta_{i,v}$ denotes the probability of the outcome v. The following assertion must hold

$$\forall_{i \in I} \Sigma_{v \in Vi} \theta_{i,v} = 1.0.$$

Let F be a set of observed facts. For each fact $f \in F$, E_f denotes the set of explanations. Let $e \in E_f$ be an explanation. The probability of e is the product of the probabilities of all the switches in the explanation:

$$\theta_e = \prod_{msw(i,v) \in e} (\theta_{i,v})$$

The probability of fact f is the sum of the probabilities of all its explanations:

$$\theta_f = \sum_{e \in E_f} (\theta_e)$$

The *log likelihood* of fact f is defined as $\ln(\theta_f)$. For each explanation $e \in E_f$, let $\delta_{i,v}(e)$ denote the number of occurrences of the switch $msw(i,v)$ in e. Figure 1 shows the EM algorithm. The algorithm repeats the estimation until the likelihood of the observed facts becomes stable.

procedure em(F) **begin**
  initialize ϵ to a small positive number;
  foreach $i \in I, v \in V_i$ initialize $\theta_{i,v}$;
  $\lambda^1 = \sum_{f \in F}(\ln(\theta_f))$; /* initial likelihood */
  repeat
    $\lambda^0 = \lambda^1$;
    foreach $i \in I, v \in V_i$
      $\eta_{i,v} = \sum_{f \in F}(\frac{\sum_{e \in E_f} (\theta_e \times \delta_{i,v}(e))}{\theta_f})$
      /* expected count of $msw(i,v)$ */
    foreach $i \in I, v \in V_i$
      $\theta_{i,v} = \frac{\eta_{i,v}}{\sum_{v' \in Vi}(\eta_{i,v'})}$
    $\lambda^1 = \sum_{f \in F}(\ln(\theta_f))$;
  until $\lambda^1 - \lambda^0 < \epsilon$
end

Figure 1: The EM algorithm

The use of the term $\eta_{i,v}$, which estimates the number of occurrences of the switch $msw(i,v)$ that contribute to the observed facts, is essential in the algorithm. The probability of $msw(i,v)$ is estimated as the ratio of its count to the count of all the outcomes of the switch.

$$\theta_{i,v} = \frac{\eta_{i,v}}{\sum_{v' \in Vi}(\eta_{i,v'})}$$

For our illustrative example, the algorithm converges in a few iterations. If only `direction(left)` is observed, then the estimated probability of `head` is close to 1.0 and that of `tail` is close to 0.0; if `direction(left)` and `direction(right)` each occupy half of the observed facts, then the estimated distribution is close to uniform. For more complicated programs, more iterative steps are required to obtain a stable estimation.

3 Improvements of the Implementation

The interpreters and the EM learning algorithm presented in the previous section are naive and inefficient. The number of explanations for a set of observed facts may be exponential. Therefore, it is expensive to find explanations and it is also expensive to go though the explanations to estimate the probabilities of the switches in the explanations. In this section, we propose several techniques for improving the implementation, especially the learning algorithm.

3.1 Explanation Graphs

It is not hard to notice that explanations differ from each other by only a small number of switches. Just as it is important to factor out common sub-expressions in evaluating expressions, it is important to factor out common switches among explanations. Actually, a logic program provides a natural structure for factoring out common switches. Instead of considering explanations as lists of switches, we consider explanations as a graph.

An *explanation path* for a fact H is defined as $(H \rightarrow B_g \ \& \ B_s)$ where B_g is a set of facts and B_s is a set of switches. H is called the *root* of the path. An explanation path corresponds to an instance of a clause where B_g is the set of probabilistic subgoals, and B_s the set of switches in the body. An *explanation tree* for a fact consists of a set of explanation paths that have the fact as the root. The root of the paths is also called the root of the tree. An explanation tree corresponds to an instance of a predicate. An *explanation graph* consists of a set of explanation trees whose roots are all distinct.

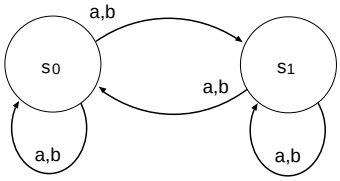


Figure 2: An example HMM.

Consider, for example, the following program that represents the two-state HMM[3] in Figure 2,

```
hmm(L,N) :-
   msw(init,Si),
   hmm(1,N,Si,L).

% Current state is S, current position is I.
hmm(I,N,S,[]) :- I>N,!.
hmm(I,N,S,[C|L]) :-
   msw(out(S),C),
   msw(tr(S),NextS),
   I1 is I+1,
   hmm(I1,N,NextS,L).

values(init,[s0,s1]).
values(out(_),[a,b]).
values(tr(_),[s0,s1]).
```

The predicate `hmm(L,N)` analyses or generates a string `L` of length `N`. The explanation graph for `hmm([a,b,a],3)` is shown in Figure 3.

It is assumed that explanation graphs are acyclic, i.e., a fact cannot be used to explain the fact itself. This assumption, however, does not rule out left recursion. Consider, for example, the following CFG rule,

```
s(I,J):-s(I,I1),a(I1,J).
```

Although `s(I,J)` and `s(I,I1)` are variants as subgoals, they are instantiated to different instances and thus no fact is used to explain the fact itself.

3.2 Constructing Explanation Graphs Using Tabling

If goals were treated independently in constructing explanation graphs, the computation would still be exponential in general. Recall the explanation graph in Figure 3. The size of the graph is $O(N \times S)$ where N is the length of the string and S is the size of the largest sample space. If shared goals in different paths, such as the two underlined ones, are considered only once, then it takes only linear time to construct the explanation graph.

[3] An HMM is a probabilistic automaton in which the selections of the initial state, output symbols, and transitions on the symbols are all probabilistic.

```
hmm([a,b,a],3)
   → hmm(1,3,s0,[a,b,a]) & msw(init,s0)
   → hmm(1,3,s1,[a,b,a]) & msw(init,s1)
hmm(1,3,s0,[a,b,a])
   → hmm(2,3,s0,[b,a]) & msw(tr(s0),s0), msw(out(s0),a)
   → hmm(2,3,s1,[b,a]) & msw(tr(s0),s1), msw(out(s0),a)
hmm(1,3,s1,[a,b,a])
   → hmm(2,3,s0,[b,a]) & msw(tr(s1),s0), msw(out(s1),a)
   → hmm(2,3,s1,[b,a]) & msw(tr(s1),s1), msw(out(s1),a)
hmm(2,3,s0,[b,a])
   → hmm(3,3,s0,[a]) & msw(tr(s0),s0), msw(out(s0),b)
   → hmm(3,3,s1,[a]) & msw(tr(s0),s1), msw(out(s0),b)
hmm(2,3,s1,[b,a])
   → hmm(3,3,s0,[a]) & msw(tr(s1),s0), msw(out(s1),b)
   → hmm(3,3,s1,[a]) & msw(tr(s1),s1), msw(out(s1),b)
hmm(3,3,s0,[a])
   → hmm(4,3,s0,[]) & msw(tr(s0),s0), msw(out(s0),a)
   → hmm(4,3,s1,[]) & msw(tr(s0),s1), msw(out(s0),a)
hmm(3,3,s1,[a])
   → msw(tr(s1),s0), msw(out(s1),a)
   → msw(tr(s1),s1), msw(out(s1),a)
```

Figure 3: The explanation graph for `hmm([a,b,a],3)`.

Tabling or memoization [Tamaki and Sato, 1986; Warren, 1992; Zhou *et al.*, 2000] can used to avoid redundant computations. The idea of tabling is to memorize the answers to subgoals and use the answers to resolve subsequent variant subgoals. The table area is global and answers stored in it can survive over backtracking. Therefore, variant subgoals can share answers regardless where they occur in execution. They can occur in the same execution path or different paths.

The following gives an interpreter for constructing the explanation graph for a goal.

```
expls(G):-
   expl(G,_,[],_,[]),fail.
      %backtrack to find all paths
expls(G).

expl((G1,G2),Bg,BgR,Bs,BsR):-!,
   expl(G1,Bg,Bg1,Bs,Bs1),
   expl(G2,Bg1,BgR,Bs1,BsR).
expl(msw(I,V),Bg,Bg,[mse(I,V)|Bs],Bs):-!,
   values(I,Values), % sample space is Values
   member(V,Values).
expl(G,[G|Bg],Bg,Bs,Bs):-
   prob_predicate(G),!,
      %G is a probabilistic predicate
   expl_prob_goal(G).
expl(G,Bg,Bg,Bs,Bs):-
   call(G).

:-table expl_prob_goal/1.
expl_prob_goal(G):-
   clause(G,Body),
   expl(Body,Bg,[],Bs,[]),
   add_to_database(path(G,Bg,Bs)).
```

The `expl(G,Bg,BgR,Bs,BsR)` is true if `Bg-BgR` is the list of probabilistic subgoals and `Bs-BsR` is the list of switches in `G`. For each probabilistic subgoal `G`, the `expl_prob_goal(G)` finds the explanation paths for `G`. The predicate `expl_prob_goal/1` is tabled. So variant probabilistic subgoals share explanation paths. The `add_to_database(path(G,Bg,Bs))` adds the path to the database if the path is not there yet.

The naive EM learning algorithm is reformulated such that it works on explanation graphs. Since explanation graphs are acyclic, it is possible to sort the trees in an explanation graph based on the calling relationship in the program. The refined algorithm is able to exploit the hierarchical structure to propagate probabilities over sorted explanation graphs efficiently.

3.3 Compilation

The interpreter presented above is inefficient since it introduces an extra level of interpretation. The interpreter version of the PRISM system is used in debugging programs. For learning from a large amount of data, it is recommended that the compiler version be used. The PRISM compiler translates a program into a form that facilitates the construction of explanation graphs.

Let `p(X1,...,Xn):-B` be a clause in a probabilistic predicate. The compiler translates it into:

```
expl_p(X1,...,Xn):-
   B',
   add_to_database(path(p(X1,...,Xn),Bg,Bs)).
```

where `B'` is the translation of `B`, `Bg` is the list of probabilistic subgoals in `B'`, and `Bs` is the list of switches in `B`. For each subgoal `G` in `B`, if `G` is `msw(I,V)`, then it is translated into `values(I,Values),member(V,Values)`. Otherwise, it is copied to `B'`, renaming each predicate `p` to `expl_p`. The translated predicate is declared as a tabled predicate, so explanation trees need to be constructed only once for variant subgoals.

For example, the predicate

```
hmm(I,N,S,[]) :- I>N,!.
hmm(I,N,S,[C|L]) :-
   msw(out(S),C),
   msw(tr(S),NextS),
   I1 is I+1,
   hmm(I1,N,NextS,L).
```

is translated into:

```
:-table expl_hmm/4.
expl_hmm(I,N,S,[]) :- I>N,!.
expl_hmm(I,N,S,[C|L]) :-
   values(out(S),Values1), % msw(out(S),C),
   member(C,Values1),
   values(tr(S),Values2), % msw(tr(S),NextS)
   member(NextS,Values2),
   I1 is I+1,
   expl_hmm(I1,N,NextS,L),
   add_to_database(path(hmm(I,N,S,[C|L]),
         [hmm(I1,N,NextS,L)],
         [msw(out(S),C),
         msw(tr(S),NextS)])).
```

Table 1: Learning times for a corpus (seconds).

# sentences	all-solution-search	EM learning
1000	268	2022
1500	445	3938
2000	855	5542

Notice that no path is added to the database for the first clause since the body does not contain switches nor probabilistic subgoals.

4 Experience

The PRISM system has been fully implemented in B-Prolog, a CLP system that supports tabling. The tabling system in B-Prolog was first implemented in 1999 [Zhou *et al.*, 2000] and was recently re-implemented to meet the requirements of PRISM. The new implementation inherits the linear tabling idea, and incorporates new strategies and optimization techniques for fast computation of fixpoints [Zhou and Sato, 2003]. As a tabling system for parsing, B-Prolog is twice as fast as and consumes an order of magnitude less stack space than XSB, a Prolog system developed at SUNY Stony Brook.

The current version of PRISM is, to our knowledge, the first of its kind that can support real-world applications. Several application projects are going on at the moment [Sato and Zhou, 2003]. One of the projects is to use PRISM to learn probabilities of the Japanese grammar rules from corpora. Table 1 shows the times spent in learning from various numbers of sentences on Windows XP (1.7GHz CPU, 760M RAM). The first phase of learning, i.e., finding explanations has improved significantly thanks to the adoption of the new tabling system in B-Prolog. The EM learning phase dominates the learning time now. In the current version, explanation graphs are represented as Prolog terms. The EM learning phase can be improved if better data structures are used.

5 Related Work

PRISM was first designed by Sato [Sato, 1995] who proposed a formal semantics, called *distribution semantics*, for logic programs with probabilistic built-ins, and derived an EM learning algorithm for the language from the semantics. The need for structural explanations was envisioned in [Sato and Kameya, 2001], but this paper presents the first serious implementation of the EM learning algorithm that works on explanation graphs.

Poole's abduction language [Poole, 1993] incorporates Prolog and Bayesian networks, in which probability distributions are given as *joint declarations*. Muggleton's stochastic logic language [Muggleton, 1996] is an extension of PCFG where clauses are annotated with probabilities. In both languages, probability distributions are specified by the users, and learning from sample data is not considered.

Non-logic based languages have also been designed to support statistical modeling (e.g., [Pfeffer *et al.*, 1999; Ramsey and Pfeffer, 2002]). The built-in function `choose` in the stochastic lambda calculus [Ramsey and Pfeffer, 2002] is similar to `msw` in PRISM, which returns a value from the

sample space randomly. Non-logic languages do not support nondeterminism. Therefore, it would be difficult to devise an EM like learning algorithm for these languages.

Tabling shares the same idea as dynamic programming in that both approaches make full use of intermediate results of computation. Using tabling in constructing explanation graphs is analogous to using dynamic programming in the Baum-Welch algorithm for HMM [Rabiner, 1989] and the Inside-Outside algorithm for PCFG [Baker, 1979].

6 Concluding Remarks

This paper has presented an efficient implementation of PRISM, a language designed for statistical modeling and learning. The implementation is the first serious one of its kind that integrates logic programming and statistical reasoning/learning. The high performance is attributed to several techniques. One is to adopt explanation graphs rather than flat explanations in learning and use tabling to construct explanation graphs. Another technique is compilation. Programs are compiled into a form that facilitates searching for all solutions.

Acknowledgement

Thanks go to Shigeru Abe for his help in implementing the refined EM learning algorithm. Part of the work by Neng-Fa Zhou was conducted while he visited Tokyo Institute of Technology in the summer of 2002.

References

[Baker, 1979] J. K. Baker. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, 1979.

[Castillo *et al.*, 1997] Enrique Castillo, Jose Manuel Gutierrez, and Ali S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, New York, 1 edition, 1997.

[Dempster *et al.*, 1976] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, pages 1–38, 1976.

[Muggleton, 1996] S. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.

[Pearl, 1987] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, Inc., 1987.

[Pfeffer *et al.*, 1999] Avi Pfeffer, Daphne Koller, Brian Milch, and Ken T. Takusagawa. Spook: A system for probabilistic object-oriented knowledge representation. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 541–550, S.F., Cal., July 30–August 1 1999. Morgan Kaufmann Publishers.

[Poole, 1993] David Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recoginition. *Proceedings of the IEEE*, 77:257–286, 1989.

[Ramsey and Pfeffer, 2002] Norman Ramsey and Avi Pfeffer. Stochastic lambda calculus and monads of probability distributions. In *Conference Record of POPL'02: The 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 154–165, Portland, Oregon, January 16–18, 2002.

[Sato and Kameya, 2001] A. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, pages 391–454, 2001.

[Sato and Zhou, 2003] T. Sato and N.F. Zhou. A new perspective of PRISM. In *IJCAI Workshop on Learning Statistical Models from Relational Data*, page 7 pages, 2003.

[Sato, 1995] Taisuke Sato. A statistical learning method for logic programs with distribution semantics. In Leon Sterling, editor, *Proceedings of the 12th International Conference on Logic Programming*, pages 715–730, Cambridge, June 13–18 1995. MIT Press.

[Stirzaker, 1994] D. Stirzaker. *Elementary Probability*. Cambridge University Press, 1 edition, 1994.

[Tamaki and Sato, 1986] Hisao Tamaki and Taisuke Sato. OLD resolution with tabulation. In Ehud Shapiro, editor, *Proceedings of the Third International Conference on Logic Programming*, Lecture Notes in Computer Science, pages 84–98, London, 1986. Springer-Verlag.

[Warren, 1992] D. S. Warren. Memoing for logic programs. *Comm. of the ACM, Special Section on Logic Programming*, 35(3):93, March 1992.

[Wetherell, 1980] C. S. Wetherell. Probabilistic languages: A review and some open questions. *ACM Computing Surveys*, 12(4):361–379, December 1980.

[Zhou and Sato, 2003] N.F. Zhou and T. Sato. Efficient fixpoint computation in linear tabling. In *Fifth ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming*, page to appear, 2003.

[Zhou *et al.*, 2000] Neng-Fa Zhou, Yi-Dong Shen, Li-Yan Yuan, and Jia-Huai You. Implementation of a linear tabling mechanism. *Lecture Notes in Computer Science*, 1753:109–123, 2000.

Anna Atramentov
Ph.D. student
Computer Science Department
Iowa State University

RESEARCH STATEMENT

My current research interests are focused on designing and implementing accurate and efficient algorithms for learning from relational databases.

There have been several approaches proposed in the literature for knowledge discovery in this setting. One of them, called multi-relational data mining, exploits structured query language (SQL) to gather the information needed for constructing classifiers (e.g., decision trees) from multi-relational data. This approach has several advantages. Firstly, it explicitly exploits the semantics of the data stored in relational tables. Secondly, it uses database primitives for efficient storing and querying the data.

We have implemented MRDTL -- a multi-relational decision tree learning algorithm proposed in this framework. Our experiments with this algorithm revealed that the execution of some queries was a major bottleneck in terms of the running time of the algorithm. Hence, we have developed an approach for significantly speeding up some of the most time consuming components of this algorithm, and other algorithms based on the multi-relational data mining framework, thus, enhancing the applicability of multi-relational data mining algorithms.

My future work is aimed at:

- Developing and incorporating techniques for handling missing values and noise in multi-relational data mining algorithms.
- Developing and incorporating more sophisticated pruning techniques in MRDTL algorithm.
- Developing and incorporating techniques for ontology-guided variants of multi-relational data mining algorithms.
- More extensive experimental evaluation of the multi-relational data mining algorithms on real world data sets.

Using Probabilistic Relational Models for Collaborative Filtering

William H. Hsu **Prashanth Boddhireddy** **Roby Joehanes**

Laboratory for Knowledge Discovery in Databases, Kansas State University
234 Nichols Hall, Manhattan, KS 66506
{bhsu | pbo8844 | robbyjo}@cis.ksu.edu http://www.kddresearch.org

Abstract

This research summary describes some work in progress on using graphical models to represent relational data in computational science portals such as *myGrid*. The objective is to provide a integrative *collaborative filtering* (CF) capability to users of data, metadata, source code, and experimental documentation in some domain of interest. Recent systems such as *ResearchIndex* / *CiteSeer* provide collaborative recommendation through citation indexing, and systems such as *SourceForge* and the Open Bioinformatics project provide similar tools such as content-based indexing of software. Our current research aims at learning *probabilistic relational models* (PRMs) from data in order to support intellignet retrieval of data, source code, and experimental records. We present a system design and a précis of a test bed under development that applies PRM structure learning and inference to CF in repositories of bioinformatics data and software.

Keywords: probabilistic relational models, collaborative filtering, information retrieval, source code repositories, structure learning

1 INTRODUCTION

Collaborative filtering is the problem of analyzing the content of an information retrieval system and actions of its users, to predict additional topics or products a new user may find useful. Developing this capability poses several challenges to machine learning and reasoning under uncertainty. The research described in this summary addresses the problem of formulating tractable and efficient problem specifications for probabilistic learning and inference in this framework. It describes an approach that combines learning and inference algorithms for relational models of semi-structured data into a domain-specific collaborative filtering system. Recent systems such as *ResearchIndex* / *CiteSeer* have succeeded in providing some specialized but comprehensive indices of full documents. The collection of user data from such digital libraries provides a test bed for the underlying IR technology, including learning and inference systems. The authors are therefore developing two research indices in the areas of bioinformatics (specifically, functional genomics) and software engineering (digital libraries of source codes for computational biology), to experiment with machine learning and probabilistic reasoning software recently published by the authors and a collaborative filtering system currently under development.

The overall goal of this research program is to develop new computational techniques for discovering *relational and constraint models* for domain-specific collaborative filtering from scientific data and source code repositories, as well as use cases for software and data sets retrieved from them. The focus of this project is on statistical evaluation and automatic tuning of algorithms for learning graphical models of uncertain domains from such data. These include probabilistic representations, such as *Bayesian networks* and *decision networks*, that have recently been applied to a wide variety of problems in intelligent information retrieval and filtering. The primary contribution of this research shall be the novel combination of algorithms for learning the structure of relational probabilistic models with existing techniques for constructing relational models of metadata about computational science experiments, data, and programs. The technical objectives center around statistical experiments to evaluate this approach on data from the domains of *gene expression modeling* and *indexing of bioinformatics repositories*.

1.1 Rationale

Recent systems such as *ResearchIndex* / *CiteSeer* [LGB99] have succeeded in providing cross-indexing and search features for specialized but comprehensive **citation** indices of full documents. The indexing technologies used by such systems, as well as the general-purpose algorithms such as *Google PageRank* [BP98] and *HITS* [KL99], have several advantages: They use a *simple conceptual model* of document webs. They require little specialized knowledge to use, but organize and present hits in a way that allows a knowledgeable user to select relevant hits and build a collection of interrelated documents quickly. They are extremely popular,

encouraging users to submit sites to be archived and corrections to citations, annotations, links, and other content. Finally, some of their content can be automatically maintained.

Despite these benefits, systems such as *ResearchIndex* have limitations that hinder their direct application to IR from bioinformatics repositories:

- **Over-generality:** Citation indices and comprehensive web search engines are designed for the generic purpose of retrieving all individual documents of interest, rather than collections of data sets, program source codes, models, and metadata that meet common thematic or functional specifications.
- **Over-selectivity:** Conversely, IR systems based on keyword or key phrase search may return fewer (or no) hits because they check titles, keywords, and tags rather than semi-structured content.
- **Lack of explanatory detail:** A typical user of an integrated collaborative filtering system has a specific experimental objective, whose requirements he or she may understand to varying degree depending upon his or her level of expertise. The system needs to be able to **explain relationships** among data, source codes, and models in the context of a bioinformatics experiment.

1.2 Objectives and Hypothesis

How can we achieve the appropriate balance of generality and selectivity? How can we represent inferred relationships among data entities and programs, and explain them to the user? Our thesis is:

> *Probabilistic representation, learning, and reasoning are appropriate tools for providing domain-specific collaborative filtering capability to users of a scientific computing repository, such as one containing bioinformatics data, metadata, experimental documentation, and source codes.*

Toward this end, we are developing *DESCRIBER*, a research index for consolidated repositories of **computational genomics resources**, along with machine learning and probabilistic reasoning algorithms to refine its data models and implement collaborative filtering. The unifying goal of this research is to advance the automated extraction of **graphical models of use cases** for computational science resources, to serve a user base of researchers and developers who work with genome data and models. We present recent results from our own work and related research that suggest how this can be achieved through a novel combination of probabilistic representation, algorithms, and high-performance data mining not previously applied to collaborative filtering in bioinformatics. Our project shall also directly advance gene expression modeling and intelligent, search-driven reuse in distributed software libraries.

2 CF IN COMPUTATIONAL SCIENCES

2.1 Collaborative Filtering Objectives

We seek to take existing ontologies and minimum information standards for computational genomics and create a refined and elaborated data model for decision support in retrieving data, metadata, and source codes to serve researchers. A typical collaborative filtering scenario using a domain-specific research index or portal is depicted in **Error! Reference source not found.** 1. We now survey background material briefly to explain this scenario, then discuss the methodological basis of our research: development of learning and inference components that take records of use cases and queries (from web server logs and forms) and produce decision support models for the CF performance element.

As a motivating example of a computational genomics experiments, we use gene expression modeling from microarray data. DNA hybridization *microarrays*, also referred to as *gene chips*, are experimental tools in the life sciences that make it possible to model interrelationships among genes, which encode instructions for production of proteins including the *transcription factors* of other genes. Microarrays simultaneously measure the expression level of thousands of genes to provide a "snapshot" of protein production processes in the cell. Computational biologists use them in order to compare snapshots taken from organisms under a control condition and an alternative (e.g., *pathogenic*) condition. A microarray is typically a glass or plastic slide, upon which DNA molecules are attached at up to tens of thousands of fixed locations, or *spots*. Microarray data (and source code for programs that operate upon them) proliferate rapidly due to recent availability of chip makers and scanners.

A major challenge in bioinformatics is to discover gene/protein interactions and key features of a cellular system by analyzing these snapshots. Our recent projects in computational genomics focus on the problem of automatically extracting gene regulatory dependencies from microarray data, with the ultimate goal of building simulation models of an organism under external conditions such as temperature, cell cycle timing (in the yeast cell), photoperiod (in plants), etc. Genomes of model organisms, such as *S. cerevisiae* (yeast), *A. thaliana* (mouse ear cress or *weed)*, *O. sativa* (rice), *C. elegans* (nematode worm), and *D. melanogaster* (fruit fly), have been fully sequenced. These have also been annotated with the *promoter* regions that contain binding sites of *transcription factors* that regulate gene

expression. Public repositories of microarray data such as the *Saccaromyces* Genome Database (SGD) for yeast have been used to develop a comprehensive catalog of genes that meet analytical criteria for certain characteristics of interest, such as *cell cycle regulation* in yeast. We are using SGD data and a synthesis of existing and new algorithms for learning Bayesian networks from data to build robust models of regulatory relationships among genes from this catalog. Most data resources we plan to use in developing *DESCRIBER* are in the public domain, while some are part of collaborative work with the UK *myGrid* project (Goble).

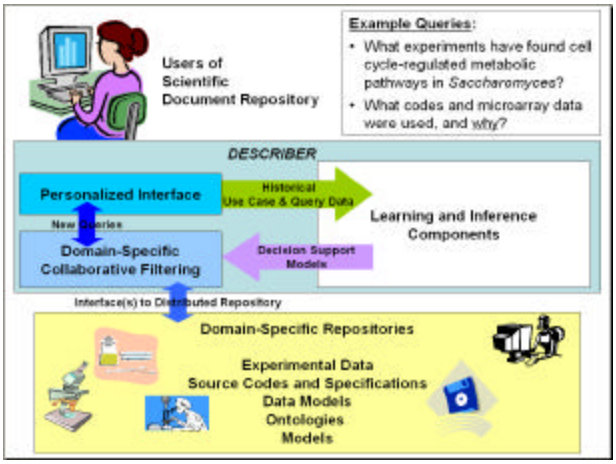


Figure 1. Design overview of DESCRIBER

The next two figures depict our design for *DESCRIBER*. Figure 2 is the block diagram for the overall system, while Figure 3 elaborates Module 1 as shown in the lower left hand corner of Figure 3. Our current and continuing research focuses on algorithms that perform the learning, validation, and change of representation (inductive bias) denoted by Modules 2 and 4. We choose probabilistic relational models as a representation because they can express constraints (cf. Figure 1) and capture uncertainty about relations and entities. We hypothesize that this will provide more flexible generalization over use cases. We have recently developed a system for Bayesian network structure learning that improves upon the *K2* [CH92] and *Sparse Candidate* [FLNP00] algorithms by using combinatorial optimization (by a genetic algorithm) to find good topological orderings of variables. Similar optimization wrappers have been used to adapt problem representation in supervised inductive learning for classification, using decision trees and instance-based learning.

Other relevant work includes *BioIR*, a digital library for bioinformatics and medical informatics whose content is much broader than that of this test bed for genome analysis. *BioIR* emphasizes phrase browsing and cross-indexing of text and data repositories rather than experimental metadata and source codes. Other systems such as *CANIS*, *SPIDER*, and *OBIWAN* also address intelligent search and IR from bioinformatics digital libraries, emphasizing categorization of text documents. We view the technologies in these systems as complementary and orthogonal to our work because of this chief difference.

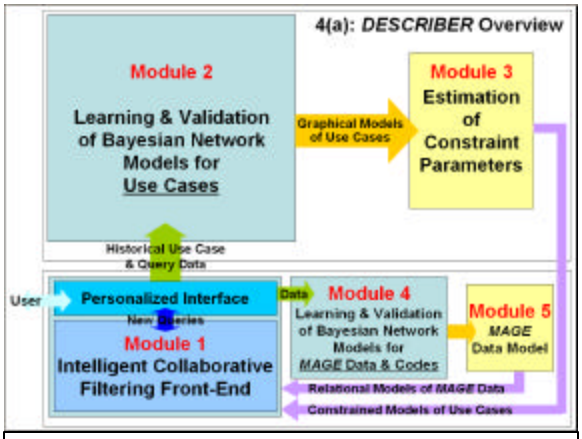


Figure 2. DESCRIBER system

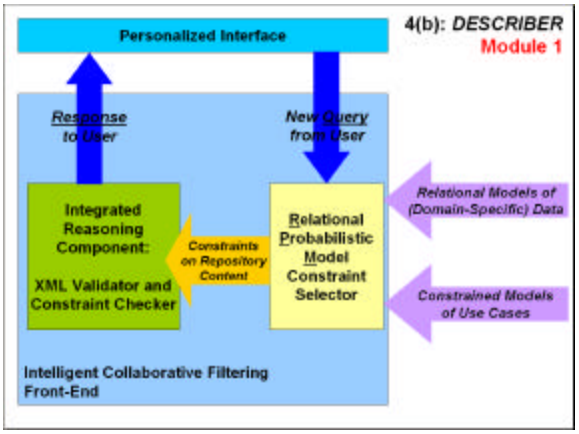


Figure 3. Collaborative filtering component of DESCRIBER

3 LEARNING BN STRUCTURE

3.1 Classifier System for Learning BN Structure

Learning the structure, or causal dependencies, of a graphical model of probability such as a Bayesian network (BN) is often a first step in reasoning under uncertainty. In many machine learning applications, it is therefore referred to as a method of *causal discovery* [PV91]. Finding the optimal structure of a BN from data has been shown to be *NP*-hard [HGC95], even without considering latent (unobserved) or irrelevant (extraneous) variables. Therefore, greedy *score-based* algorithms

[FG98] have been developed to provide more efficient structure learning at an accuracy tradeoff. In this paper we examine a general shortcoming of greedy structure learning – sensitivity to variable ordering – and develop a genetic algorithm to mitigate this problem by searching the permutation space of variables [HH98] using a probabilistic inference criterion as the fitness function.

We make the case in this paper that the probabilistic inference performance element, **in the absence of a known gold standard network** or any explicit constraints, can provide the feedback needed to search for a good ordering. We then derive a heuristic based on validation by inference (exact inference [LS88, Ne90] for small networks, approximate inference by stochastic sampling [CD00] for larger ones). Our primary objective is inferential accuracy *using* the learned structure.

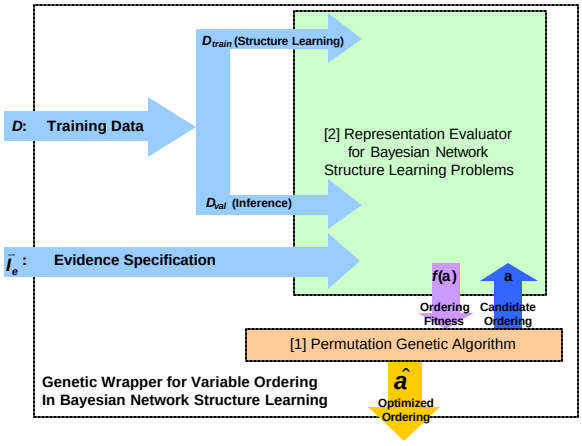


Figure 4. System Design Overview.

Toward this end, we have developed a *genetic wrapper*, similar to a classifier system [BGH89], to search the space of variable orderings in score-based structure learning. This wrapper adapts a composite fitness measure used in other wrappers based upon best-first search [KJ97] and automatically tunes parameters of the learning system [HL99] such as the ordering of input variables. We present the system shown in Figure 1, a genetic algorithm-based wrapper [CS96, RPG+98, Hs03], and show how it provides a parallel stochastic search mechanism for inferential loss-minimizing variable orderings. We demonstrate that, used in tandem with *K2*, it produces structures whose loss under importance sampling is nearly as low as any found by exhaustive enumeration of orderings. Finally, we discuss how this wrapper provides a flexible method for tuning *representation biases* [Mi97] in Bayesian network structure learning using different fitness criteria.

Consider a typical probabilistic reasoning environment, as shown in Figure 2, where structure learning [A] is a first step. The input to this system includes a set D of training data vectors $\mathbf{x} = (x_1, \ldots, x_n)$ each containing n variables. If the structure learning algorithm is greedy, an ordering $\mathbf{a}$ on the variables may also be given as input. The structure learning component of this system produces a graphical model $B = (V, E, \Theta)$ that describes the dependencies among X_i, including the conditional probability functions. The inferential performance element [B] of this system takes B and a new data set D_{test} of vectors drawn from the desired inference space, where only a subvector $\mathbf{E}$ of $\mathbf{X} = (X_1, \ldots, X_n)$ is observable, and infers the remaining unobserved values $\mathbf{X} \setminus \mathbf{E}$. We denote the indicator bit vector for membership in $\mathbf{E}$ by $\mathbf{I_e}$. The performance criterion f is the additive inverse of the (inferential or utility) loss of [B].

4 CONTINUING WORK

Our current research focuses on structure learning of relational models by adapting traditional score-based search algorithms for flat graphical models [Pe03] *and* constrain-based structure search over hierarchical models.

Entity and reference slot uncertainty present new challenges to PRM structure learning. Three of the questions that we are looking into are:

1. *How much relational data is needed?* How can we estimate the sample complexity of PRMs under specified assumptions about entity existence and reference slot distributions?
2. *What constraint-based approaches can be usedf?* Learning reference slot and entity structure in PRMs presents a task beyond flat structure learning.
3. *Can this cut down on the amount of data to learn the low-level model (versus the flat version)?* How can we establish and test sufficient conditions for conditional independenc, and context-specific independence, in PRMs?

5 References

[BGH89] L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier Systems and Genetic Algorithms. *Artificial Intelligence*, 40:235-282, 1989.

[BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems, 30*(1-7):107-117.

[CD00] J. Cheng and M. J. Druzdzel. AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. Journal of Artificial Intelligence Research (JAIR), 13:155-188, 2000.

[CH92] G. F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning, **9**(4):309-347, 1992.

[CS96] K. J. Cherkauer and J. W. Shavlik. Growing Simpler Decision Trees to Facilitiate Knowledge Discovery. In *Proceedings of the Second International Conference of Knowledge Discovery and Data Mining (KDD-96)*, Portland, OR, August, 1996.

[FG98] N. Friedman and M. Goldszmidt. *Learning Bayesian Networks From Data*. Tutorial, American

National Conference on Artificial Intelligence (AAAI-98), Madison, WI. AAAI Press, San Mateo, CA, 1998.
[FLNP00] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, Using Bayesian networks to analyze expression data. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology (RECOMB 2000)*, ACM-SIGACT, April 2000.
[HGC95] D. Heckerman, D. Geiger, and D. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197-243, Kluwer, 1995.
[HH98] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*. Wiley-Interscience, New York, NY, 1998.
[HL99] G. Harik and F. Lobo. *A parameter-less genetic algorithm.* Illinois Genetic Algorithms Laboratory technical report 99009, 1999.
[Hs03] W. H. Hsu. Control of Inductive Bias in Supervised Learning using Evolutionary Computation: A Wrapper-Based Approach. In J. Wang, *editor*, *Data Mining: Opportunities and Challenges*, p. 27-54. IDEA Group Publishing.
[Kl99] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM, 46*(5):604-632.
[KJ97] R. Kohavi and G. H. John. Wrappers for Feature Subset Selection. *Artificial Intelligence, Special Issue on Relevance, 97*(1-2):273-324, 1997.
[Mi97] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.
[LGB99] S. Lawrence, C. L. Giles, and K. Bollacker Digital libraries and autonomous citation indexing. *IEEE Computer, 32*(6):67-71.
[LS88] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B* 50, 1988.
[Ne90] R. E. Neapolitan. Probabilistic Reasoning in Expert Systems: Theory and Applications. Wiley-Interscience, New York, NY, 1990.
[Pe03] B. B. Perry. *A Genetic Algorithm for Learning Bayesian Network Adjacency Matrices from Data.* M.S. thesis, Department of Computing and Information Sciences, Kansas State University, 2003.
[PV91] J. Pearl and T. S. Verma, A theory of inferred causation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*. Morgan Kaufmann, San Mateo, CA, 1991.
[RPG+97] M. Raymer, W. Punch, E. Goodman, P. Sanschagrin, and L. Kuhn, Simultaneous Feature Extraction and Selection using a Masking Genetic Algorithm, In *Proceedings of the 7th International Conference on Genetic Algorithms*, pp. 561-567, San Francisco, CA, July, 1997.

Statement of Interest

Gwendolyn E. Campbell, Ph.D., Amy E. Bolton, Wendi L. Bolton
NAVAIR Orlando Training Systems Division
Orlando, FL 32826-3275
{gwendolyn.campbell, amy.bolton , wendi.buff}@navy.mil

For the past five years we have been conducting a program of research investigating alternative techniques for identifying and interpreting patterns in human performance data collected during sessions in training simulators. Our three primary research questions have been:

(1) How closely can each technique fit human performance data?
(2) How well does each technique identify patterns that are consistent with the subjective reports of the human participants?
(3) How effective is training feedback that is based on a critique of the performance patterns derived with each technique?

To date, we have investigated the application of multiple linear regression, nonlinear regression, fuzzy logic, and classification and regression trees (CART). We are currently investigating discrete choice analysis and several "fast and frugal" algorithms (ala Gigerenzer).

A number of the potential topics of this workshop would be of interest to us. Obviously, we are interested in learning about new techniques and gaining insight into the representational power of different techniques. In addition, within a training simulator, events unfold over time, and thus we need to find effective ways to incorporate background knowledge (or history information) when modeling performance data. Finally, our work has a strong focus on the application of models to support training goals. It is our hope that identifying performance patterns in data collected from a training simulator should help the trainer determine a student's strengths and weaknesses, and thus support the development of adaptive and tailored feedback, instruction and scenario exercises. We would welcome the opportunity to talk to other researchers who are interested in similar applications.

References for our Research

Campbell, G. E., Buff, W. L., & Bolton, A. E. (in preparation). Viewing training through a fuzzy lens. To appear in A. Kirlik (Ed.), Working with Technology in Mind: Brunswikian Resources for Cognitive Science & Engineering. Oxford University Press.

Campbell G. E. & Bolton, A. E. (2003) Mathematical Models of Human Decision Making: Rational, Fuzzy, or Fast and Frugal? Paper submitted to the 47th Annual Conference of the Human Factors and Ergonomics Society.

Bolton, A. E., Buff, W. L., & Campbell, G. E. (2003). Faster, cheaper, and "just as good"? A comparison of the instructional effectiveness of three HBRs that vary in development requirements. To be presented at the 12th Conference on Behavior Representation in Modeling and Simulation, Scottsdale, AZ, May 12-15, 2003.

Campbell, G. E., Buff, W. L. & Bolton, A. E. (2002) Traditional mathematical modeling applied to human prioritization judgments: Quicksand for the unwary modeler. Presented at the Eleventh Conference on Computer Generated Forces. May 7th-9th, 2002, Orlando, Florida.

Campbell, G. E., Buff, W. L., Bolton, A. E. & Holness, D. O. (2001). The application of mathematical techniques for modeling decision-making: Lessons learned from a preliminary study. In E. M. Altman, A. Cleermans, C. D. Schunn & W. D. Gray (Eds.), Proceedings of the Fourth International Conference on Cognitive Modeling (pp. 49-54). Mahwah, NJ: Lawrence Erlbaum Associates.

Bolton, A. E., Holness, D. O., Buff W. L., & Campbell, G. E. (2001). The application of mathematical models in training systems: A viable approach to cognitive modeling? *Proceedings of the Tenth Conference on Computer Generated Forces*, 497-505.

Holness, D. O., Buff, W. L., Bolton, A. E. & Campbell, G. E. (2001). Delivering feedback in a training system using mathematical modeling techniques. Poster presented at the 16th annual meeting of the Society of

Industrial and Organizational Psychology, San Diego, CA.

Campbell, G. E., Buff, W. L. & Bolton, A. E. (2000). The diagnostic utility of fuzzy system modeling for application in training systems. Proceedings of the XIVth Triennial Congress of the International Ergonomics Association and the 44th Annual Meeting of the Human Factors and Ergonomics

Campbell, G. E., Buff, W. L., Rhodenizer, L. G. & Dorsey, D. W. (1999). Decision making in a tactical setting: Crisp or fuzzy reasoning? Paper presented at the Third International Conference on Cognitive and Neural Systems, Boston, MA.

Campbell, G. E., Rhodenizer, L. G., & Buff, W. L. (1999). A comparison of two mathematical modeling approaches to produce cognitive feedback. Paper presented at the 43th annual meeting of the Human Factors and Ergonomics Society. Houston, Texas.

Buff, W. L. & Campbell, G. E. (1999). Subjective Decision-Making Reports Correspond More Closely to Fuzzy Models of Decision Data. Paper presented at Annual Conference of the American Psychological Society, Denver: CO.

Statement of Interest

Bruce D'Ambrosio
CleverSet Inc.
dambrosi@cleverset.com

I first began studying extensions to traditional graphical probabilistic models in 1997. At that time I was working on applying Bayesian networks to the DARPA Dynamic Databases Program (DDB). The background domain knowledge was clearly modular, and organized around entities and their relationships. This information was often noisy and uncertain, and so a probabilistic extension to traditional AI frame-based representation languages seemed an obvious solution. Over the next few years we (IET, Inc) developed a probabilistic frame-based knowledge-representation language (JPF java probabilistic frames) that supports uncertainty over the existence and type of frame instantiations and relationships among them, as well as over primitive attributes. Of particular interest in that project was the problem of specifying role expectations, distributions over attributes of reference slot fillers. We successfully applied JPF in both DDB and in the DARPA Information Assurance and CyberPanel programs. Issues in representation, dynamic construction, and inference in probabilistic relational models remain a continuing interest.

More recently, my groups at OSU (Oregon State University) and CleverSet, Inc have been studying the discovery of dynamic statistical relational models from data. This arises in two contexts. The submitted paper describes applications in scientific data analysis. Here the primary goal is the learned model itself, and the insights gained from examining it. A second focus, the topic of a UAI submission in preparation, is the discovery of dynamic models of on-line user behavior. Web log entries are not IID. A complex relational structure exists among users, sessions, page requests, and web-site content. This is especially true for ecommerce sites, where content dimensions include both functional dimensions (is a page a search page, a directory page, a product page, &) and product attribute dimensions. In this application, the primary focus is on application of learned models for real-time interpretation of user intent. The relationship between the PRM language we use in these projects at CleverSet, JPF, the variations on PRM reported in the literature, and other statistical relational models, in terms of expressivity, convenience for knowledge engineering, and support for efficient inference, are open issues of great interest.

Relational Learning for Securities Market Regulation

Position Paper for the IJCAI 2003 Workshop

Henry G. Goldberg*

NASD (National Association of Securities Dealers), 9513 Key West Avenue, Rockville, MD 20850
GoldberH@NASD.com

Introduction and Background

Over the past six years, NASD's Market Regulation department has built and operates two major "break detection" systems [Senator 2002] – the Advanced Detection System (ADS) and Securities Observation, News Analysis, and Regulation (SONAR) – for surveillance of the Nasdaq and several other markets. These systems rely for their effectiveness on the detection of instances of scenarios of regulatory interest – episodes in market activity where some violation may have occurred – many of which comprise relationships among transactions, market participants, securities, issuers, and other subject entities. I will discuss these systems in a bit more detail and then describe some of the kinds of scenarios for which statistical learning would be most beneficial.

Advanced Detection System (ADS)

ADS monitors trades, quotations, and orders in the Nasdaq, Over the Counter (OTC), and Nasdaq-Liffe (futures) stock markets to identify patterns and practices of behavior of potential regulatory interest. [Kirkland 1999] ADS has been in operational use at NASD since summer 1997 by several groups of analysts, processing roughly 25 million transactions per day, generating several thousand breaks per day. More important, it has greatly expanded surveillance coverage to new areas of the market and to many new types of behavior of regulatory concern. It's technology has been expanded to surveillance of the corporate and municipal bond markets and to NASD's new Alternative Display Facility. ADS combines detection and investigative components in a single system which supports multiple regulatory domains and which share the same market data. ADS makes use of a variety of AI techniques, including visualization, pattern recognition, and data mining, in support of the activities of regulatory analysis, alert and pattern detection, and knowledge discovery. ADS relies on a rule pattern matcher and a time-sequence pattern matcher. Data and market visualizations allow analysts to see the market context of breaks and temporal relationships of events in large amounts of data.

Temporal/Sequence Relationships in ADS

ADS relies heavily upon heuristic, manually coded patterns describing temporal sequences of market transactions. These patterns are input to a sequence matcher which finds instances of the patterns in databases of market transactions. The sequence matcher algorithm is similar to a regular expression matcher. It maintains a list of potential match states. At each step, a row is fetched and a new state is started for each pattern. Existing states are advanced if they match data constraints on the current transaction. When a state reaches the end of a pattern, it is a match. The sequence matcher may be in increasing or decreasing time order depending on whether the triggering event for the sequence occurs before or after the other necessary conditions. In a single pass, multiple tables may be scanned for several patterns concurrently. The sequence pattern language uses a syntax and precedence similar to the C programming language.

The sequence match has several problems. It is extremely brittle, in the sense that patterns and data constraints must be very carefully drawn not to inadvertently exclude a potentially valued match. A single failed match kills the entire chain. As a result, break detection errors are usually allowed to run heavily towards the false positives. Pattern discovery is limited to a semi-automated, iterative process in which patterns are carefully refined in an attempt to achieve the desired results and error rates.[Senator 2000] However, this refinement is, of necessity, haphazard and incomplete in its ability to model the variability in the data. Finally, there is a critical need to detect what market analysts call a "pattern and practice" – a set of similar or related matches from which one may infer intention violation of rules.

It is likely that statistical modeling can help to address all three problems. Models which produce a likelihood that an episode belongs to the modeled population are less brittle. Pattern refinement through statistical method would be more consistent and a comprehensive in dealing with data variability. And a model which describes a population of sequence episodes is a promising step towards defining "pattern and practice" detection.

*The author of this paper is an employee of NASD. The views expressed herein are those of the author and do not represent an official policy statement of NASD.

Securities Observation, News Analysis, and Regulation (SONAR)

SONAR was developed by NASD to monitor the Nasdaq, Over the Counter (OTC), and Nasdaq-Liffe (futures) stock markets for potential insider trading and fraud through misrepresentation. [Goldberg 2003] SONAR has been in operational use at NASD since December 2001, processing approximately 10,000 news wires stories and SEC filings, evaluating price/volume models for 25,000 securities, and generating 40-50 alerts (or "breaks") per day for review by several groups of regulatory analysts and investigators. SONAR makes use of several AI and statistical techniques, including NLP text mining, statistical regression, rule-based inference, uncertainty, and fuzzy matching. Sonar combines these enabling technologies in a system designed to deliver a steady stream of high-quality breaks to the analysts for further investigation. Additional components including visualization, text search agents, and flexible displays add to the system's utility.

Entities, Relationships, and Events

SONAR mines news wire stories and SEC filings for entities such as companies which issue securities, company officers, brokers, the securities themselves, regulatory bodies such as the FDA which have an impact on stock values, and others. It also finds material events: product announcements, earnings reports, mergers and acquisitions, etc. Finally, SONAR mines for relationships both explicit and implicit among the entities and events. The results of the text mining stage are contained in the top-level predicates output by a linguistic rule-base used by SONAR NLP component (from ClearForest). These entities, relationships, and events form particular episodes or scenarios, with specific identifiers and values which may be incompletely mined. Learning statistical models of these episodes would improve detection, especiallyin dealing with stories where the "components" of a scenarios are not all present.

News Stream Segmentation

Insider Trading is defined as trading upon inside information of a "material" nature – information which a reasonable investor would take as a reason to buy or sell a security. Thus, two crucial events in an insider trading break are the appearance of material news and a movement in the market in response to it. It is critical that SONAR is able, therefore, to determine when a news item is material, but also when it is first made public. The drawing of relationships among entities and events mined from several news stories is currently performed by a fairly simple template match. But, clearly, news is re-written, expended upon, and interpreted. Any failure of this match will "create" a new trigger for an insider trading break. Membership in the same model, drawn from a broad population of multiple story events, seems to be better way to detect truly "new" news.

Misrepresentation Fraud

Fraud by misrepresentation is another critical target activity of SONAR. While we currently mine for several dozen "flags", likely indicators of stocks which are being falsely touted, much more could be done with the ability to draw comparisons across stories and sources (e.g. compare an announcement of $50M dollars in contracts with an SEC filing indicating the company has a staff of 2 with no assets.) Linking such evidence across text sources and learning statistical models of misrepresentation seems to be a promising approach.

Break Detection and Fraud

Break Detection Systems are powerful tools for detecting errors, violations, or other anomalous conditions and activities. [Senator 2002] However, they are limited to the immediate activities which they find in the input data stream. Background knowledge, aggregation of detection over a priori identifiers (brokers, issuers, etc.) can start to draw a picture of an underlying intentional pattern and practice. Without powerful but tractable models of populations of breaks, we are limited to counts and percentages as a decision tool for investigators. As target activities become more complex and varied, and as the cost of regulation continues to rise, NASD feels increased need for such models to cull and derive the greater benefit from its break detection systems.

References

Goldberg, Henry G, Kirkland, James D., Lee, Dennis, Shyr, Ping, and Thakker, Dipak, "The NASD Securities Observation, News Analysis & Regulation System (SONAR)," *presented at IAAI-2003, Acapulco, Mexico.*

Kirkland, James D., Senator, Ted E., Hayden, James J., Dybala, Tomasz, Goldberg, Henry G., and Shyr, Ping, "The NASD Regulation Advanced Detection System (ADS)," *AI Magazine* 20(1):55-67, 1999.

Senator, Ted E., "Ongoing Management and Application of Discovered Knowledge in a Large Regulatory Organization: A Case Study of the Use and Impact of NASD Regulation's Advanced Detection System (ADS)," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00)*, pp. 44-53, ACM, 2000.

Senator, Ted E. and Goldberg, Henry G., "Break Detection Systems." in W. Kloesgen and J. Zytkow (eds.), *Handbook of Knowledge Discovery and Data Mining*, pp. 863-873, Oxford University Press, 2002.

Multi-view Learning: A Special Case of Relational Learning

Ion Muslea
muslea@ics.uci.edu
Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425

Abstract

In this research statement, I begin by briefly describing the connection between relational and multi-view learning, and then I summarize my main results on multi-view learning. Finally, I explain how these results are relevant in the more general framework of relational learning.

1 Introduction

By definition, in *multi-view* learning tasks [Blum and Mitchell, 1998] the target concept can be independently learned within different *views*; i.e., from disjoint sets of features, each of which is sufficient to learn the target concept. For instance, Web pages can be classified *either* based on words in the documents *or* based on words in the hyperlinks pointing to them [Blum and Mitchell, 1998]; similarly, voice recognition can be performed based on *either* sound *or* leap motion features [de Sa and Ballard, 1998].

Existing multi-view learners [Blum and Mitchell, 1998; Nigam and Ghani, 2000; Ghani, 2002; Pierce and Cardie, 2001] are *semi-supervised* algorithms (i.e., they learn from a few labeled and many unlabeled examples) that work as follows: first, they use a few labeled examples to learn a hypothesis in each view; then they use a large number of unlabeled examples to bootstrap the views from each other. Such algorithms, which typically perform statistical learning in each view, were successfully applied to a variety of real-world domains, from Web page classification [Blum and Mitchell, 1998] to statistical parsing [Sarkar, 2001] to named entity classification [1999].

Note that a domain with two views (such as the ones mentioned above) can be seen as a relational domain that is defined over the binary relation "*IsDescriptionOfSameObject()*"; that is, for each instantiation *IsDescriptionOfSameObject*(e_1, e_2), the entities e_1 and e_2 represent the description of *the same object* in the two views.[1] At the same intuitive level, multi-view bootstrapping algorithms can be seen as simplified versions of statistical relational learners such as the ones described in [Taskar *et al.*, 2002; Getoor *et al.*, 2001].

[1] The idea generalizes in a straightforward manner to k-ary relationships.

2 Results in Multi-view Learning

In my recent work, I focused primarily on minimizing the amount of labeled data required for learning in multi-view domains. In order to reduce the need for labeled data, I used active learning algorithms that detect and ask the user to label only the most informative examples in a domain.

First, I introduced Co-Testing [Muslea *et al.*, 2000; Muslea, 2002; Muslea *et al.*, 2003], which is a family of multi-view active learners that are based on the idea of learning from mistakes. Co-Testing starts with a few labeled and many unlabeled examples, and it uses the few labeled examples to learn a hypothesis in each view. Then these hypotheses are applied to the unlabeled examples, and the user is asked to label one of the examples on which the views make different predictions (if two views disagree, one of them is guaranteed to make a mistake); finally, the entire process is repeated for a number of iterations. Under assumptions [Muslea, 2002], I proved that this "learning from mistakes" strategy leads to faster convergence than competing approaches.

I also showed that existing multi-view learners perform unreliably if the views are inadequate (e.g., if the views are highly correlated, or if they are insufficiently expressive to accurately learn the target concept). To cope with this problem, I introduced two complementary solutions. First, by interleaving active and semi-supervised multi-view learning, I obtained a novel multi-view learner that has a robust behavior over a wide spectrum of domains that have inadequate views [Muslea *et al.*, 2002a]. Second, I introduced a meta-learning algorithm that is first trained on several *solved* learning tasks and then predicts whether or not the views are "sufficiently adequate" for solving new, unseen learning tasks [Muslea *et al.*, 2002b].

I have evaluated these three novel algorithms on a variety of real-world domains, from information extraction and text classification to advertisement removal and discourse tree parsing. These experiments show that compared with existing state-of-the-art approaches, my algorithms require up to two orders of magnitude fewer training examples.

3 Multi-view and Relational Learning

All the three issues discussed above in the context of multi-view learning also apply to statistical relational learning:

- **active learning:** how can one exploit the domain's relational structure in order to detect and ask the user to label only the most informative examples? In a recent paper [Muslea *et al.*, 2003], I have taken a first step in this direction by proposing a framework for active learning with *strong* and *weak* views (i.e., views in which one can learn the target concept **or** a concept that is strictly more general/specific than the target one, respectively).
- **robust learning:** how can one minimize the effect of feature correlation and features that provide contradictory evidence? I conjecture that the idea of interleaving active and semi-supervised learning [Muslea *et al.*, 2002a] can be successfully applied to statistical relational learning.
- **meta-learning:** for one new, unseen learning task, how can one know whether the relational domain is "sufficiently rich" to benefit from statistical relational learning (rather then simply "flattening" the dataset and performing learning in the resulting propositional, feature-vector dataset)? I believe that training a meta-learner from tasks that are labeled as sufficiently or insufficiently "rich" for relational learning would represent an interesting starting point for such an investigation.

References

[Blum and Mitchell, 1998] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT-98*, pages 92–100, 1998.

[Collins and Singer, 1999] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the Empirical NLP and Very Large Corpora Conference*, pages 100–110, 1999.

[de Sa and Ballard, 1998] Virginia de Sa and Dana Ballard. Category learning from multi-modality. *Neural Computation*, 10(5):1097–1117, 1998.

[Getoor *et al.*, 2001] Lise Getoor, Eran Segal, Ben Taskar, and Daphne Koller. Probabilistic models of text and link structure for hypertext classification. In *Proceedings of the IJCAI-2001 Workshop on Text Learning: Beyond Supervision*, 2001.

[Ghani, 2002] Rayid Ghani. Combining labeled and unlabeled data for multiclass text classification. In *Proceedings of the 19th International Conference on Machine Learning (ICML-2002)*, pages 187–194, 2002.

[Muslea *et al.*, 2000] Ion Muslea, Steven Minton, and Craig Knoblock. Selective sampling with redundant views. In *Proccedings of the National Conference on Artificial Intelligence AAAI-2000*, pages 621–626, 2000.

[Muslea *et al.*, 2002a] Ion Muslea, Steven Minton, and Craig Knoblock. Active + Semi-supervised Learning = Robust Multi-view Learning. In *Proceedings of the International Conference on Machine Learning ICML-2002*, pages 435–442, 2002.

[Muslea *et al.*, 2002b] Ion Muslea, Steven Minton, and Craig Knoblock. Adaptive view validation: A first step towards automatic view detection. In *Proceedings of the International Conference on Machine Learning ICML-2002*, pages 443–450, 2002.

[Muslea *et al.*, 2003] Ion Muslea, Steven Minton, and Craig Knoblock. Learning with strong and weak views: a case study on wrapper induction. In *Proccedings of the International Joint Conference on Artificial Intelligence IJCAI-2003*, 2003.

[Muslea, 2002] Ion Muslea. *Active Learning with Multiple Views*. PhD thesis, Department of Computer Science, University of Southern California, 2002.

[Nigam and Ghani, 2000] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of Information and Knowledge Management*, pages 86–93, 2000.

[Pierce and Cardie, 2001] David Pierce and Claire Cardie. Limitations of co-training for natural language learning from large datasets. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP-2001)*, pages 1–10, 2001.

[Sarkar, 2001] Anoop Sarkar. Applying co-training methods to statistical parsing. In *Proceedings of the 2nd Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, pages 175–182, 2001.

[Taskar *et al.*, 2002] Ben Taskar, P. Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)*, 2002.

Statistical Relational Learning at U Penn

Alexandrin Popescul
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
popescul@cis.upenn.edu

Dean P. Foster
Department of Statistics
University of Pennsylvania
Philadelphia, PA 19104
foster@gosset.wharton.upenn.edu

Lyle H. Ungar
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
ungar@cis.upenn.edu

We do statistical relational learning by incrementally extracting data from a relational database, and computing features of that data which are then used in a classical discriminative statistical model component. Candidate features for the model are generated by a structured search in the space of relational database queries and selected using statistical information criteria. The structuring of the search space is inspired by techniques in inductive logic programming (ILP), but the use of statistical modeling relaxes the necessity of limiting the search space to logical expressions. We use a rich feature space that includes clusters, which can be generated incrementally and used to augment the basic relational schema. Current areas of research include determining optimal model selection criteria for use in this setting where an infinite sequence of features can be incrementally generated and the use of intelligent search heuristics to focus search on more promising subspaces.

A growing number of machine learning applications of high interest involves the analysis of data which is both noisy and is of complex relational structure. This dictates a natural choice in such domains: the use of statistical rather than deterministic modeling and relational rather than propositional representation [Popescul *et al.*, 2002]. Classical statistical learners provide powerful modeling component but are often limited to a "flat" file propositional domain representation where potential features are fixed-size attribute vectors. Often the manual process of preparing such attributes is costly and not obvious when more complex regularities are involved. We are developing a methodology which combines the strengths of classical statistical models with the higher expressivity of features automatically generated from a relational database.

Our interest in statistical relational learning developed while working on modeling in CiteSeer[1], an online digital library of computer science papers. CiteSeer contains a rich set of relational data, including citation information, the text of titles, abstracts and documents, author names and affiliations, conference or journal names. Applications we have addressed include: i) prediction in social networks, e.g. link prediction: given two papers estimate whether they cite each other [Popescul and Ungar, 2003], ii) document classification, modeling of more complex features than traditional word counts improves classification accuracy [Popescul *et al.*, 2003]. We are planning to apply statistical relational learning in bioinformatics domains, in particular for prediction of protein-protein interactions.

Figure 1 highlights the main components of our learning setting. Two main processes—relational feature generation and statistical modeling—are coupled into a single loop. Knowing which features have been selected by the statistical modeler allows the query generation component to guide its search, focusing on promising subspaces of the feature space.

Our statistical relational learning approach has several key features which distinguish it from either pure probabilistic modeling or inductive logic programming.

- We assume an application domain in which there are many entities connected by many relations (e.g. a patient database in a hospital), in which complex features (e.g. a set of patients clustered by the similarity of the symptoms, and treated by doctors working in the same clinic) are highly predictive of outcomes of interest (e.g. expected stay in the hospital). In such areas, it is generally not feasible to build a large generative model (e.g. a PRM) of the world, and a more focussed exploration of the space of possible relations is needed.
- Our search in the query space is an instance of propositionalization, as proposed in the inductive logic programming community; however, the use of statistics rather than logic allows the formulation of rich feature spaces, extending far beyond boolean-valued features. This richer search space can include statistical summaries or aggregates, more

[1] `http://citeseer.org/`

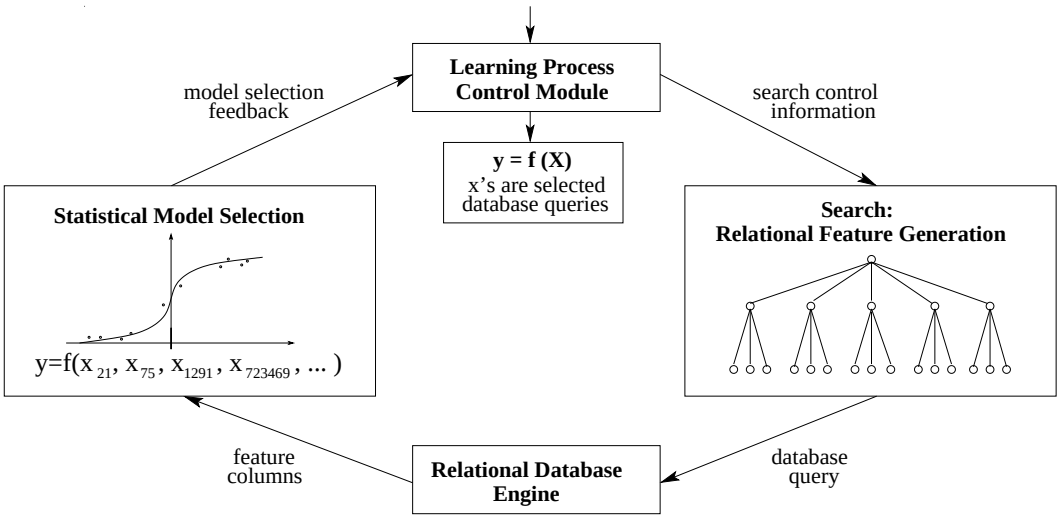


Figure 1: Learning process diagram. The search in the space of database queries involving one or more relations produces feature candidates one at a time to be considered by the statistical model selection component. The process results in a statistical model where each selected feature is the evaluation of a database query encoding a predictive data pattern in a given domain.

expressive substitutions through nesting of intermediate aggregates (e.g., how many times does this publication cite the most cited author in conference to which it was submitted?) A key question is how best to define the search space and how to control the search space complexity and search space bias.

- We use clustering to extend the set of relations generating new features. Clusters improve modeling of sparse data, improve scalability, and produce richer representations [Foster and Ungar, 2002]. New clusters can be derived using the same features used in the statistical modeling. For example, one can cluster words based on co-occurrence in documents, giving "topics", or authors based on the number of papers they have published in the same venues, giving "communities." Once clusters are formed, they represent new relationships (e.g. `on_topic_3(paper1798)` or `in_community_5(author7)`), which can be added to the relational database schema, and then used interchangeably with the original relations.
- Learning takes place with an exponential number of potential feature candidates, only relatively few of which are expected to be useful. Feature selection methods recently derived by statisticians give promising results for handling this potentially infinite stream of features with only a finite set of observations.
- Our formulation supports sophisticated procedures for determining which subspaces of the query space to explore. Intelligent search techniques which combine the relational structure of the data, feedback from the feature selection algorithm, and other information such as sampling from feature subspaces to determine their promise will help scale to truly large problems.
- We use relational database management systems (RDMSs) and SQL rather than Prolog. Most real data lie in RDMSs, which have specified schema and meta-information which we can use. RDMSs also incorporate decades of work on optimization, providing better scalability.

References

[Foster and Ungar, 2002] Dean Foster and Lyle Ungar. A proposal for learning by ontological leaps. In *Proceedings of Snowbird Learning Conference*, Snowbird, Utah, 2002.

[Popescul and Ungar, 2003] Alexandrin Popescul and Lyle H. Ungar. Statistical relational learning for link prediction. In *Proc. of the Workshop on Learning Statistical Models from Relational Data at IJCAI-2003*, 2003.

[Popescul *et al.*, 2002] Alexandrin Popescul, Lyle H. Ungar, Steve Lawrence, and David M. Pennock. Towards structural logistic regression: Combining relational and statistical learning. In *Proc. of the Workshop on Multi-Relational Data Mining at KDD-2002*, 2002.

[Popescul *et al.*, 2003] Alexandrin Popescul, Lyle H. Ungar, Steve Lawrence, and David M. Pennock. Statistical relational learning for document mining. Computer and Information Sciences, University of Pennsylvania, 2003. *http //www.cis.upenn.edu/~popescul/publications.html*.

Taisuke Sato
Dept. of Computer Science,
Tokyo Institute of Technology, Japan

My research has been (and will be) revolving around the integration of logic (programming) and (statistical) learning, including (1)probabilistic semantics for logic programs as an extension of the traditional fixed point semantics, (2) developing a symbolic-statistical modeling language PRISM with tabling and EM learning for defining arbitrarily complex distributions by programs, (3) a divide-and-conquer approach to first order decision trees for inducing logic programs, using First Order Compiler, a deterministic program synthesis algorithm based on unfold/fold program transformation, and (4) applying symbolic-statistical modeling techniques to new fields such as game playing.

Research Statement

Neng-Fa Zhou

May 29, 2003

My research interests are in the design and implementation of programming languages and systems, including logic, constraint, and object-oriented programming languages and systems. Modern programming languages have been progressing towards higher levels of abstractions that allow programmers to model the complex world of objects and their interactions, and to describe more about "what" to do than about "how" to do it. We have witnessed a gradual switch from structured programming to object-oriented programming in certain sectors of industry. Constraint languages allow programmers to describe the relations that must be satisfied, not the algorithms to satisfy them. The declarative feature of the new generation programming languages can greatly improve the productivity of software development and its maintainability.

The higher a language's description level is, the bigger the gap is between its execution model and the real machines. My research has focused on how to narrow the gap by designing efficient abstract machines and developing smart compilation techniques. I have authored thirty papers in my area and have served on the program committees of several international conferences. I have developed the B-Prolog system, a fast constraint logic programming system which has several thousands users world-wide in both academia and industry: many universities (for example, UCLA, Harvard, Science University of Tokyo, Nangoya Institute of Technology, Shizuoka University, Fukuoka Institute of Technology, The University of Mississippi, and The University of Alberta) use the system in their logic and constraint programming courses; and several companies, including Microsoft, Toyota in Japan, and JCatalog in Germany, have purchased commercial licenses of the system.

My more recent research effort has been on the design and implementation of (1) linear tabling, a method for efficient evaluation of recursive logic programs; (2) action rules, a new language for constructing interactive systems including constraint propagators; (3) CGLIB, a constraint-based high-level graphics library for constructing dynamic and interactive graphics; and (4) PRISM, a statistical modeling and learning system (joint work with Professor Taisuke Sato of Tokyo Institute of Technology).

Linear Tabling

Tabling is a technique that can get rid of infinite loops for bounded-term-size programs and redundant computations in the execution of recursive logic programs. The main idea of tabling is to memorize the answers to subgoals and use the answers to resolve their variant descendents. Tabling not only is useful in the problem domains that motivated its birth, such as program analysis, parsing, deductive database, and theorem proving, but also has been found essential in several other problem domains such as model checking, learning, and data mining.

Early resolution mechanisms proposed for tabling such as OLDT rely on suspension and resumption of subgoals to compute fixpoints. I, together with Yi-Don Shen, invented a new resolution framework called linear tabling which has received considerable attention because of its simplicity, ease of implementation, and good space efficiency. The idea of linear tabling is to use depth-first iterative deepening rather than suspension to compute fixpoints. Linear tabling is still immature compared with OLDT and a great of potential remains to be exploited. The objective of this project is to analyze possible strategies and work out a cutting-edge implementation of the tabling method.

Action Rules (AR)

The lack of facilities for programming "active" sub-goals that can be reactive to the environment has been considered one of the weaknesses of logic programming. AR is an extension of Prolog designed to overcome this weakness. A program in AR consists of a sequence of action rules. Each action rule specifies a pattern for agents, an action that the agents can carry out, and an event pattern for events that can activate the agents. AR combines goal-oriented execution model of logic programming with even-driven execution model. This hybrid execution model facilitates constructing interactive systems such as constraint propagators, interactive user interfaces, and multi-agent systems.

This project aims at an efficient implementation of AR. It investigates implementation techniques including memory architectures and activation mechanisms for agents, compilation of agent conditions, parallel execution of action rules, and optimization techniques tailored to constraint propagators.

CGLIB: A Constraint-Based Graphics Library

The widespread use of window systems has made a graphics package indispensable for any programming languages. The objective of this project is to develop a high-level constraint-based and efficient graphics library for B-Prolog. The library provides primitives for creating and manipulating graphical objects and a set of constraints that facilitates the specification of

the layouts of objects. It also provides constructs and built-in events for creating agents and programming interactions among agents or between agents and the user. A prototype of the library has been developed but several challenging tasks remain to be conducted. One task is to extend the base shapes to include three-dimensional ones, and another task is to develop effective heuristics and constraint reasoning techniques for fast packing.

PRISM: A Statistical Modeling and Learning Tool

PRISM, a logic-based language designed originally by Taisuke Sato, supports statistical modeling and learning. In PRISM, tabling is used to find explanations for sample data and the EM learning algorithm is adopted to learn probabilities. As a symbolic statistical modeling language, PRISM subsumes several specific statistical tools such as HMM (Hidden Markov Models), PCFG (Probabilistic Context Free Grammars) and discrete Bayesian networks. Compared with numeric models where mathematical formulas are used, PRISM offers incomparable flexibility by allowing the use of arbitrary logic programs to describe probability distributions. PRISM can be used in many areas such as language processing, decision-making, bio-informatics, and game theory where randomness or uncertainty is essential. This project is closely related to the project on linear tabling. In addition to an efficient tabling method, this project also entails the development of PRISM-specific compilation and optimization techniques.

Aggregation-Based Feature Invention for Relational Learning

Claudia Perlich and Foster Provost
Stern School of Business
44 West 4th Street
New York, NY 10012
cperlich@stern.nyu.edu,fprovost@stern.nyu.edu

Preliminary version of publication with identical title presented at KDD-2003

Keywords: Relational Learning, Aggregation, Feature Invention

Abstract

The field of relational data mining/learning, which traditionally has been dominated by logic-based approaches, has recently been extended by "relational upgrades" of traditional learning methods such as naive Bayes, Baysian networks and decision trees. One aspect inherent to all such methods of model induction from relational data is the construction of features via the aggregation of sets. The theoretical part of this work (1) presents an ontology of relational concepts of increasing complexity, (2) derives classes of aggregation operators that are needed to learn these concepts, and (3) classifies relational domains based on relational schema characteristics such as cardinality. We then present a new class of aggregation functions that are particularly well suited for relational classification and class probability estimation. The empirical part of this paper demonstrates on a noisy business domain the effects of different aggregation methods on predictive system performance. The results suggest that more-complex aggregation methods can significantly increase generalization performance and that, in particular, task-specific aggregation can simplify relational prediction tasks into well-understood propositional learning problems.

1 Motivation and Introduction

Relational learning has attracted significant attention due to the expressive power of relational models and the techniques' ability to incorporate relational background knowledge. Until recently, relational learning research has been dominated by Inductive Logic Programming (ILP)[15]. Other approaches include distance-based methods[7], binary propositionalization[10], SQL-based numeric aggregation[8], and upgraded propositional learners such as rule learners [11], Structural Logistic Regression [16], Relational Decision Trees [5] and Probabilistic Relational Models (PRM)[9]. The aggregation of sets of related objects into single attributes is an essential component of relational model induction, and has significant impact on generalization performance for domains with important 1-to-n relationships. However, with the exception of [8], aggregation has received little direct attention. Aggregation methods can be characterized along a number of dimensions including the underlying calculus (numeric or logical), the cardinality of the set, and the complexity of the objects being aggregated (atomic values or feature vectors, single-type or multi-type objects).

The objective of this paper is to shed new light on the role of aggregation methods in relational learning. We present a hierarchy of classes of relational concepts; different aggregation operators are appropriate for different classes. We also evaluate relational learners on a noisy business domain and draw conclusions about the applicability and performance of different aggregation operators—including some novel ones. For this paper we have chosen the relational database formalism for expressing relational data and concepts. However, the ideas and methods carry over directly to learning from graph or first-order-logic representations.

The paper is organized as follows: Section 2 presents an ontology of increasingly more complex relational concepts, and discusses the complexity of domains and the relationship between domain properties and concept complexity. Section 3 presents an overview of existing aggregation methods, their limitations, and systems that apply them. We also propose a novel target-dependent aggregation method. The subsequent empirical study in section 4 compares a number of aggregation methods on a relational business domain and shows evidence of the superiority of more complex methods (viz., target-dependent set aggregation). We conclude with suggestions for future work with particular focus on more complex aggregation methods than currently are used.

2 Hierarchy of Relational Concepts

We consider a predictive (rather than clustering or unsupervised) relational learning task as finding a mapping $M : (t, RDB) \rightarrow y$ where t is a row of the target table T,[1] including a target variable y (either numeric for a regression task or categorical for classification), and RDB is a relational database containing additional tables of related background knowledge. Figure 1 shows a simple example of

[1] T is a table of traditional feature vectors, including categorical variables possibly with large numbers of possible values.

a relational database schema with three tables, the target table **Customer** with target attribute y and the background tables **Transaction**, **ReturnedItems** and **Products**, related through the keys CustomerId and ProductId. We will use this example to illustrate the examples in the following sections. The

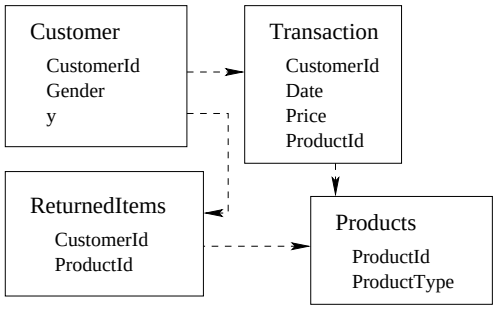


Figure 1: Transaction Database

database RDB can vary from simple to complex, in terms of the number of tables, the number of relationships between tables through shared categorical variables (*keys*), and the cardinality of those relationships (1-to-1, 1-to-n, or n-to-m).

Similar to RDBs, relational concepts can have various complexities. In this paper we adopt the view that a relational concept $(t, RDB) \rightarrow y$ is a function F including as input t and a fixed number of aggregates a_i of objects that are related to the target case through keys. In this paper we assume F to be deterministic given a vector of n aggregates, but the target observations may be noisy: $\hat{y} = F(t, A(RDB)) + \epsilon$ where aggregation function A is a mapping $RDB \rightarrow (a_0, a_1, \ldots, a_n)$.

More generally, the complexity of a relational concept is determined by

- the complexity of the relationships (e.g. cardinalities),
- the complexity of the aggregation function A,
- and the complexity of F.[2]

The complexity of the relationships is determined by the domain and the prediction task. The relative complexity of different functions F is comparatively well understood (and we have methods capable of learning very complex functions F). The complexity of aggregations, however, has received comparatively little treatment. Consider three levels of aggregation complexity:

Definition 1: A *simple aggregation* A_1 is a mapping $* \times 1 \rightarrow a$ that takes as input a bag of zero or more (deonted by $*$) atomic values (either numerical or categorical).

[2]There is an interesting tradeoff between aggregation complexity and function complexity since parts of F can be integrated into A. However, it is generally not possible to make up for lack of complexity in A through a more complex function since aggregation always involves loss of information that cannot be recovered.

Examples of simple aggregation operations for numeric values are count, mean and maximum. Typical aggregates of categorical values are the most common value (mode) or the count of the most common value.

Definition 2: A *multidimensional aggregation* A_n is a mapping $* \times N \rightarrow a$ that takes as input a bag of zero or more objects with n attributes in form of a feature vector $(x_1, ..., x_n)$.

The important difference between the use of multiple simple aggregations and a multidimensional aggregation is that the attributes in the vectors cannot be treated as independent, but must be aggregated jointly. A common case of this type is: boolean conditioning on one attribute (selection) and then a simple aggregation of a different attribute of all selected objects. More generally, a multidimensional aggregation captures any relationship between two or more attributes, for instance the slope of prices over time (explaining whether a customer is buying increasingly more expensive products). Time series data often harbor concepts where the observation clearly is dependent on a temporal field and independent aggregation would not be meaningful.

Next let us consider aggregations that cannot be achieved with a multidimenisonal aggegation or any function thereof. For example, consider finding the total value of the products that a customer has returned. This aggregation has to incorporate two bags: the products bought by the customer, along with their prices, and the products returned.

Definition 3: A *multi-type aggregation* $A_{n,m,...}$ is a mapping $* \times N, * \times M, \ldots \rightarrow a$ that takes as input two (or more) bags of objects of different types. The objects in bag one have feature vectors of length n, the objects in bag two have feature vectors of length m.

Given these definitions, we now can present a hierarchy of relational concept classes. A concept class C_A is more complex than class C_B if any concept c_b in C_B can be expressed in C_A and there are concepts in C_A that cannot be expressed in C_B. We will assume a target table T with target column y, and background tables B, C, and D that are related to T and potentially to each other via keys. A lowercase expression t denotes one row in a table T. Objects t in T and b in B are related by keys: k_{T_i,B_j} appears in T as column i and in B as column j, and is commonly a categorical variable with a large number of possible values. The operator $\pi_{B.(f,...,l)}(T \overset{\bowtie}{\underset{t_i=b_j,(1:n)}{}} B)$ denotes a left outer join $\bowtie$ of tables T and B under the condition $t_i = b_j$ and the subsequent projection of columns $f, \ldots, l$ from B. The notation 1:n (join cardinality) declares that for every value t_i there can be zero or more rows in B fulfilling the equality condition $t_i = b_j$. Given the complexity of notation we will keep the simple form of single joins; however note that it is straightforward to extend the hierarchy replacing $\pi_{B.(1,...,f)}(T \overset{\bowtie}{\underset{t_i=b_j,(1:n)}{}} B)$ by a chain of such operators joining across multiple tables $\pi_{C.(...,f)}(T \overset{\bowtie}{\underset{t_i=b_j,(1:n)}{}} B \overset{\bowtie}{\underset{b_k=c_g,(1:n)}{}} C)$.

The following list presents relational concept classes in

order of increasing complexity. For the sake of simplicity, where possible we only include one aggregation in a function F. However, the distinction between classes is not in the number of aggregations needed to express a particular concept, but rather in the complexity of the most complex aggregation used.

1. Propositional:
$y = F(t)$ or $y = F(t, (\pi_{B.(1,\ldots,f)}(T \overset{\bowtie}{\underset{t_i=b_j,(1:1)}{}} B)))$
A relationship of cardinality 1:1 returns exactly one object (feature vector) for each object in T. There is no need for aggregation and the features of objects in B can be concatenated directly to the feature vector in T. A typical case is a Customer table T and a Demographics table B that contains additional information for each customer. If the relationship has a cardinality of n:1 there will also be exactly one observation in B for each observation in T.
$y = F(t, (\pi_{B.(1,\ldots,f)}(T \overset{\bowtie}{\underset{t_i=b_j,(n:1)}{}} B)))$ An example is the abstraction hierarcy in the Product table where ProductType is an abstraction of a particular product into a class, for instance 'book'.

2. Bags of independent attributes:
$y = F(t, A_1(\pi_{B.h}(T \overset{\bowtie}{\underset{t_i=b_j,(1:n)}{}} B)), A_1(\pi_{B.k}(T \overset{\bowtie}{\underset{t_i=b_j,(1:n)}{}} B)))$
or
$y = F(t, A_1(\pi_{B.h}(T \overset{\bowtie}{\underset{t_i=b_j,(1:n)}{}} B)), A_1(\pi_{C.k}(T \overset{\bowtie}{\underset{t_i=c_j,(1:n)}{}} C)))$
The least complex relational concept class requires only simple aggregations. The object of a 1-to-n relationship may have a number of attributes, each of which can be aggregated independently. For example, simple concepts like 'the average price of products bought' fall into this category. An example that requires multiple simple aggregations from different tables is 'the proportion of products returned by the customer' using the count of the products in the Transaction table and the count of products in the ReturnedItems table for this customer. Calculating the proportion would be part of the function F, not the aggregation.

3. Bags of dependent attributes within one table:
$y = F(t, A_n(\pi_{B.(f,\ldots,l)}(T \overset{\bowtie}{\underset{t_i=b_j,(1:n)}{}} B)))$
The attributes from the objects in table B cannot be aggregated independently as before but have to be considered jointly using a multidimensional aggregation. The number of products bought on December 22^{nd} is an example that could be expressed using conditioning (on Date), selection and then a simple aggregation. A more complex example is the slope of the price over time explaining whether a customer is buying increasingly more-expensive products.

4. Multiple bags across tables:
$y = F(t, A_{n,m}(\pi_{B.(f,\ldots,l)}(T \overset{\bowtie}{\underset{t_i=b_j,(1:n)}{}} B),$
$\pi_{C.(k,\ldots,r)}(T \overset{\bowtie}{\underset{t_i=c_j,(1:n)}{}} C)))$
The total amount spent on items later returned requires information from both tables (Transactions, ReturnedItems) during the aggregation. Since the two tables may have different types they cannot simply be merged into one. Note that even if they have the same type (the joins ending in the same table) it may be important to know from which join a particular object has come. This information is lost if the results are simply merged.[3]

5. Global graph features:
$y = F(t, A_1(T \overset{\bowtie}{\underset{t_i=b_j,(1:n)}{}} B.A_{n,m}(\Omega)))$ where
$\Omega = TC(B \overset{\bowtie}{\underset{b_k=c_g,(1:n)}{}} C, \ldots, D \overset{\bowtie}{\underset{d_i=b_m,(1:n)}{}} B)$
TC stands for transitive closure. Such a global concept could for instance be a function of customer reputation. The aggregation $A_{n,m}$ for reputation may require the construction of an adjacency matrix and the calculation of its eigenvalues and eigenvectors.

It should be noted that within the five concept classes there are many sublevels of concept compexity. We will discuss some on them for level 2 throughout the paper. For instance, relational autocorrelation has been identified as a common property of relational domains[6]. Aggregations taking advantage of relational autocorrelation can be considered a special case of the single-set-value concept class, where the joins link back to the target relation and the target variable, if known, is aggregated $A_1(\pi_{T.y}(T \overset{\bowtie}{\underset{t_i=b_k,(1:n)}{}} B, \ldots, D \overset{\bowtie}{\underset{d_n=t_y,(1:n)}{}} T))$.

3 Relational Aggregation

We now discuss existing aggregation methods and how they relate to the concept classes that can be learned by different approaches. Relational learning has taken two approaches: (1) aggregation-based feature construction/invention and subsequent model estimation and (2) direct learning of a relational mapping M. Aggregation must take place in either approach; the main difference between the two is whether the aggregation is optimized jointly with the estimation of F (often rendering the aggregation more complex) or whether they are performed independently. As shown for some cases in the presentation of the ontology, there are interactions between the aggregation and the function. That notwithstanding, for the remainder of this paper we focus on the aggregation operators A, assuming the existence of some strategy for the identification of related objects[4] as well as an appropriate learner for F.

The following sections present three common approaches to aggregation in relational learning and one novel approach that combines set-distances with target-dependent aggregation.

3.1 First-order Logic

The field of relational learning for years has been dominated by Inductive Logic Programming (ILP)[15], focusing on classification tasks. These first-order-logic-based approaches search for sets of clauses that identify positive examples.

[3]The question of whether two equal-type bags can be merged prior to aggregation in combination with a simpler multidimensional aggegation depends on whether the aggregation is transitive under the bag-merging operation. The *minimum* for instance is transitive, since the minimum of the minimums of two sets is the same as the minimum of the merged set.

[4]For example, graph traversal using foreign keys as links and tables as nodes.

For instance a clause learned to characterize a rich customer might be:

RichCustomer(x)← Customer(X,Y,Z),
Transaction(X,V,P,W), P≥100

The prediction of an ILP model is positive if at least one of the clauses is true for the particular case. Binary propositionalization [19],[10] also learns sets of (first-order) clauses, but rather than using them directly for prediction it constructs binary features that are given as input to a traditional learning method (e.g., decision tree induction) to learn the function F. Both ILP and binary propositionalization use existential unification of first-order-logic clauses for aggregation. Given the tables from section 2, the example clause Customer(X,Y,Z), Transaction(X,V,P,W), P≥100 is 1 for a particular customer X if he bought a product that cost more than USD 100. The bag of products that are related to a customer is aggregated into a single binary value (0 or 1) based on the condition P≥100. The major advantage of logic-based aggregation is its ability to address all levels of complexity as outlined in section 2, including dependent bags across tables. The task of identifying customers that bought a product that was returned by another customer who bought it after 2001 can be expressed in FOL as:

Customer(X),Transaction(X,V,P,W),ReturnedItem(Y,W),
Transaction(Y,B,C,W),B≥2001

The disadvantage of logic-based aggregation is the common lack of support for numeric aggregation. In particular, it is impossible in pure logic to express that the product was returned more than 20 times. A clause can test whether the maximum of a numeric set is larger than a particular value but it can not estimate the mean or the cardinality of the set. In order for an ILP system to apply numeric aggregates they have to be declared by the user of the system as intensional background knowledge (as for instance proposed by Muggleton [14]) and only a few systems support such intensional declarations.

ILP is currently the only approach that explores concepts up to level 4. However, without the explicit numeric support through intensional background knowledge, ILP methods are severely limited in expressive power in comparison to the methods discussed below.

3.2 Simple Numeric Aggregation

A number of relational approaches including Probabilistic Relational Models (PRM)[9] and 'upgraded' propositional learners such as Relational Decision Trees [5] rely on a set of simple (mostly SQL-based) aggregation operators such as mean, min, max, count for numerical values, and proportions and most common value for categorical variables. Numeric aggregates in combination with logic-based feature construction were proposed by Knobbe et al. ([8]). These operators apply only to bags of single attributes and cannot express concepts above level 2 that require dependent aggregation.

3.3 Set Distances

Kirsten, Wrobel and Horwath [7] proposed a distance-based method for relational learning. The approach classifies objects using a k-nearest-neighbor method with a predefined relational distance metric. This metric aggregates two bags of objects related to two cases by calculating the minimum distance of all possible pairs of objects, choosing one from one bag and one from the other. The distance between two objects is the sum of squared distances for numeric values and edit distances for categorical values, normalized by the number of attributes. If an attribute is a key, rather than taking the edit distance the algorithm proceeds recursively and estimates the distance of all objects related to the current vector using that key. This form of aggregation implicitly assumes attribute independence and does not take advantage of numeric aggregates like count or average.

3.4 Target-Dependent Bag Aggregation

We now present a novel aggregation methodology for level 2 concepts that integrates the idea of vector distances and utilization of the target variable. It also illustrates the different levels of concept complexity within a class (level 2) in the concept hierarchy.

Our methodology was motivated by the observation that relational databases commonly have attributes with large numbers of possible values—and these attributes often are unsuitable for learning. A common method to aggregate a single categorical attribute with numerous values is the selection of a subset of values that appear most often and convert them into dummy variables or counts. However the most common values may not be the most predictive for a given relational learning task. We present an approach that examines distributions of values conditioned on the classes of the training cases.[5]

Definition 4: Given an arbitrary order (pairs of value v: index i) over the possible values of a categorical attribute $B.j$, a *case vector* $CV^t_{B.j}(\pi_{B.j}(T \frac{\bowtie}{t_i=bg_k,(1:n)} B))$ at position i is equal to the number of values v $(v : i)$ in the bag returned by the join and projection $\pi_{B.j}(T \frac{\bowtie}{t_i=bg_k,(1:n)} B)$ for the case t in the target table T. The bag of ProductTypes {book,CD,CD,book,DVD,book} for a specific case t under the order (watch:1,book:2,CD:3,DVD:4) would result in $CV^t = (0, 3, 2, 1)$.

Definition 5: Given an order (v:i) over the values of the attribute $B.j$, a *reference vector* $RV^c_{B.j}(\pi_{B.j}(T \frac{\bowtie}{t_i=bg_k,(1:n)} B))$ under the condition c at position i is equal to the sum of values $CV^t[i]$ for all cases t for which c was true.

Definition 6: Given an order (v:i) the *variance vector* $VV^c_{B.j}(\pi_{B.j}(T \frac{\bowtie}{t_i=bg_k,(1:n)} B))$ at position i is equal to the variance $\frac{(CV^t_{B.j}[i])^2}{N_c - 1}$ over all case t for which c was true. N_c is the number of cases for which the condition c was true.

Target-Dependent Categorical Values

Rather than selecting values that are most common across all related objects, a target-dependent approach will select categorical values that are most commonly related to positive training cases (y=1) and analogously those that are most commonly related to negative cases. To create positive dummies,

[5]The methodology is easily extended to numeric values using discretization and coding numerical values as categorical dummies.

Method	Definition
MOC	$CV[i]$ where i is the index with maximum value in unconditional reference vector RV
MOP	$CV[i]$ where i is the index with maximum value in positive reference vector $RV^{y=1}$
MON	$CV[i]$ where i is the index with maximum value in negative reference vector $RV^{y=0}$
MOD	$CV[i]$ where i is the index with maximum absolute value in vector $RV^{y=1} - RV^{y=0}$
MON	$CV[i]$ where i is the index with maximum absolute value in vector $\frac{RV^{y=1}[i]-RV^{y=0}[i]}{VV^{y=1}[i]+VV^{y=0}[i]}$

Table 1: Summary of Aggregation Methods for single categorical values

we select given $RV^{y=1}$ those values v given the order $(v : i)$ for which the $RV^{y=1}[i]$ is maximal across all entries in RV. Similarly we select those values v for which the $RV^{y=0}[i]$ is maximal. A more complex, comparative approach selects categorical values that are common for one class but not common for the other. In particular we select the values i for which the absolute value of $RV^{y=1}[i] - RV^{y=0}[i]$ is maximal. The Mahalanobis distance [12] improves over this approach by normalizing the scores by the variances before selecting the maximum:

$$\frac{RV^{y=1}[i]-RV^{y=0}[i]}{VV^{y=1}[i]+VV^{y=0}[i]}$$

Table 3.4 summarizes the five strategies to select dummies from single categorical values grouped into three groups of increasing complexity: target independent most common (first row), target-dependent most common positive or negative value (second row and third row), and the value that has the maximum difference between the positive and negative reference vectors (fourth row and fifth row).

Vector Distances

Using vector distances allows the summarization of all entries in the reference vectors, rather than using only a subset with the largest counts in the reference vectors. From each case vector VC and a reference vector RV we estimate four vector distances: edit distance (ED), Euclidean distance (EU), Mahalanobis distance (MA), and Cosine distance (COS). Since it is not clear a priori which of the distances will best capture the underlying concept, all of them are calculated and it is left to the function estimator for F to select among them.

In addition to these distances we also calculate for each of the four measures the difference between the distances to the positive and negative reference vectors:

$$EDD = ED(RV^{y=1}, CV) - ED(RV^{y=1}, CV)$$
$$EUD = EU(RV^{y=1}, CV) - EU(RV^{y=1}, CV)$$
$$COSD = COS(RV^{y=1}, CV) - COS(RV^{y=1}, CV)$$
$$MAD = MA(RV^{y=1}, CV) - MA(RV^{y=1}, CV)$$

Combining the options for distance and target conditions, we have a three-by-four matrix of vector-based aggregations in table 3.4. The vector distances can be grouped similarly into three increasingly more complex groups: target independent (first row), dependent on either positive or negative reference vector (second and third row), and dependent on the difference between the class distances (fourth row).

Reference Vector	Euclidean	Edit	Cosine	Mahalanobis
All	EU	ED	COS	MA
Positive	EUP	EDP	COSP	MAP
Negative	EUN	EDN	COSN	MAN
Positive vs. Negative	EUD	EDD	COSD	MAD

Table 2: Summary of Vector-Based Aggregation Methods

It should be noted that since these aggregations use the target to estimate features, the subsequent model can be overly optimistic about the value of the feature, which can lead to overfitting when these features are used for learning. Therefore, for the results that follow, the reference vectors, vector distances and special categorical values are estimated on 50% of the training set and the model is estimated using the other 50% of the training set.

4 Experimental Results

In this section we present results comparing the presented aggregation methods on a relational learning problem in the domain of initial public stock offerings. We include the comparative performance of four logic-based relational learners (FOIL[18], Tilde[1],Lime[13],Progol[14]) since they can express concepts of up to level 4. All other methods are within level 2. The next sections present the domain description, a brief overview of the methodology, and our results.

4.1 Domain: Initial Public Offerings

Initial public stock offerings have a unique ticker for the firm that is selling shares of their equity. An IPO is typically headed by one or occasionally two banks and supported by a number of additional banks as underwriters. The task of the bank is to put shares on the market, to set a price, and to guarantee with their experience and reputation that the stock of the issuing firm is indeed valued correctly.

The IPO domain consists of 5 tables:

- IPO(Date,Size,Price,Ticker,Exchange,SIC,Runup)
- HEAD(Ticker,Bank)
- UNDER(Ticker,Bank)
- IND(SIC,Ind2)
- IND2(Ind2,Ind)

The last two relations, IND and IND2 represent an instance of an abstraction hierarchy on SIC classifications. For example the industry code 7372 identifies the division of 'Prepackaged software'. This particular category of industry group is a member of the major group 'Business Services' with the 2 digit code 73.

In this domain, Date, Size, Price and Runup are numerical variables; Ticker, Bank, SIC, Ind2 are categorical and keys, and Ind and Exchange are simple categorical attributes. The

Method Name	Description
NO	No feature construction, only the attribtues in the IPO table
MOC	Attributes in IPO table and counts of most common categoricals (MOC)
VD	Attributes in the IPO table and vector distances EU, ED, COS, MA to unconditional reference vector
MVD	Attributes in IPO table, most common categoricals and unconditional vector distances (EU, ED, COS, MA)
MPN	Attributes in IPO table and counts of most common positive (MOP) and negative (MON) categoricals
VDPN	Attributes in the IPO table and vector distances to positive and negative reference vectors (EUP, EUN, EDP, EDN, COSP, COSN, MAP, MAN)
MVDPN	Attributes in the IPO table, most common positive (MOP) and negative (MON) categoricals, and vector distances to positive and negative reference vectors (EUP, EUN, EDP, EDN, COSP, COSN, MAP, MAN)
MD	Attributes in IPO table and counts of most common discriminative categoricals (MOD, MOM)
VDD	Attributes in the IPO table and differences of the vector distances to positive and negative reference vectors (EUD, EDD, COSD, MAD)
MVDD	Attributes in the IPO table, and counts of most common discriminative categoricals (MOD, MOM), and differences of the vector distances to positive and negative reference vectors (EUD, EDD, COSD, MAD)
AH	Attribtues in IPO and attribute Ind in table Ind2 related throught abstraction hierarchy
AC	Attributes in IPO and proportion of positive training cases (excluding the particular case) that a case was related to
LF	Logic-based features extracted from the clauses learned by FOIL

Table 3: Summary of contrasted aggregation approaches

classification task is to predict whether the offer was (would be) made on the NASDAQ exchange.

4.2 Methods

We compare the generalization performance of 4 general approaches: simple numeric aggregation, ILP, logic-based feature construction, and target-dependent set aggregation. We also constructed two other features from the relational background data: when there is an instance of an abstraction hierarchy (a sequence of n:1 joins) we include the values directly in the feature vector (AH). We also wanted to test for (and potentially take advantage of) relational autocorrelation. Therefore, we allowed joins to go back to the target table and created an "autocorrelation" aggregation (AC) representing the proportion of linked, positive training cases (excluding the particular case in question of course). Table 1 summarizes the approaches.

For the evaluation of the aggregation methods we had to implement (1) an exploration strategy that finds related objects, (2) a feature selection step to reduce the number of features, and (3) a learner that finds a model to predict the target given the aggregates. We used straightforward approaches for each of these steps.

Exploration: Given a set of tables and keys, the system constructs a graph with tables as nodes and keys as links between tables and executes a breadth-first search, starting from the target relation, over all possible exploration chains of increasing length. The exploration stops once the number of chains exceeds a stopping criterion. The second number in the size column in table 4 shows the stopping criterion (maximal number of joins) for the exploration. For each exploration chain, the system executes the corresponding join and selects all attributes from the last table joined to. It then applies the aggregation methods of varying complexity to every attribute independently. The resulting values (one for every row in the target table) are appended to the original feature vector from the target table.

Feature Selection: Once the stopping criterion is met the system selects (10 times) a subset of 10 features using weighted sampling based on estimated performance. We tried alternative methods for feature selection without much difference in performance.

Model Estimation: We used C4.5 [17] to learn the model for each of the 10 feature sets and reported the average as the final prediction. The results did not change significantly using logistic regression for the modeling.

Logic-Based Feature Construction: In order to evaluate logic-based feature construction we used the ILP system FOIL [18] to learn n FOL clauses and appended the corresponding binary features to the feature vector in the target table IPO. This methodology has been applied successfully by King [20] and by Populescul et al. [16] to text classification.

ILP: We selected four ILP system based on availability, platform independence and diversity. FOIL [18] uses a top-down, separate-and-conquer strategy adding literals to the originally empty clause until a minimum accuracy is achieved. Tilde [1] learns a relational decision tree using FOL clauses in the nodes to split the data. Lime [13] is a top-down ILP system that uses Bayesian criteria to select literals. Progol [14] learns a set of clauses following a bottom-up approach that generalizes the training examples. We did not provide any additional (intensional) backgound knowledge beyond the facts in the database. We supplied a declarative language bias for Tilde, Lime, and Progol (as required).

Evaluation: Generalization performance is evaluated in terms of classification accuracy and area under the receiver operating curve (ROC) [2]. Note that ILP systems only produce class labels but no probability scores. We therefore in-

Size	NO	MOC	VD	MVD	MPN	VDPN	MVDPN	MD	VDD	MVDD
250: 6	0.619	0.641	0.679	0.634	0.627	0.683	0.671	0.635	0.675	0.690
250: 9	0.619	0.685	0.665	0.665	0.664	0.685	0.697	0.695	0.682	**0.703**
250:12	0.619	0.674	0.655	*0.706*	0.675	*0.714*	0.694	0.659	0.697	**0.703**
500: 6	0.635	0.663	0.674	0.679	0.674	0.679	0.685	0.675	0.711	**0.741**
500: 9	0.635	0.706	0.686	0.684	0.692	0.705	*0.721*	0.725	0.697	0.737
500:12	0.635	0.689	0.689	*0.71*	0.706	0.707	0.696	0.711	**0.741**	0.739
1000: 6	0.671	0.677	0.691	0.685	0.667	0.717	0.709	0.702	0.713	0.747
1000: 9	0.671	0.705	*0.71*	0.688	0.715	*0.745*	*0.745*	0.735	0.747	0.747
1000:12	0.671	0.702	0.705	0.708	0.711	0.723	0.727	0.715	**0.767**	0.759
2000: 6	0.699	0.675	0.689	0.681	0.667	0.709	0.729	0.691	0.73	0.758
2000: 9	0.699	0.729	0.69	0.719	0.731	0.728	*0.76*	0.731	0.753	0.764
2000:12	0.699	0.715	0.709	*0.73*	0.718	0.733	0.723	0.72	**0.779**	0.758

Table 4: Classification accuracy of set aggregation methods grouped by complexity

cluded ILP only in the accuracy comparisons. All results are generalization performance on a test set of size 800 averaged over 5 runs. We refrained from including the error bars of $\pm$ one standard deviation in the table but included them in the figures.

4.3 Results

Table 2 shows the generalization performance of the set of aggregation methods as a function of training-set size and the number of joins allowed. The methods are grouped into four classes of increasing complexity: no feature construction (NO), target-independent set aggregation (MOC, VD, MVD), target-dependent set aggregation dependent on either positive or negative class (MPN, VDPN, MVDPN), and target-dependent set aggregation dependent on the difference between the positive and negative reference vectors (MD,VDD, MVDD). To help with the abbreviations (needed to make the table legible) a condensed summary of the different methods under comparison can be found in table 1. Within each class of methods in table 2, the first column presents an aggregation method that uses only single categorical aggregation, the second only vector distances, and the third both.

The best performance for each training size is highlighted in bold, and the best performance for each of the complexity classes in italics. The results show that as the complexity of the aggregation method increases, the performance increases as well. The best performance within a block is always one of the two aggregations including vector distances and using only single categorical values is almost always outperformed by vector-distance aggregation. Increasing the exploration depth (number of joins) improves performance in most cases, however the marginal effect decreases. Specifically, the increase in performance moving from 6 joins to 9 is larger than moving from 9 to 12 joins. In some cases moving from 9 to 12 joins hurts the performance for two reasons: (1) the longer the chain that relates objects to a target case, the further away the objects and the less relevant they are; (2) since features are constructed from every join, the number of features increases linearly in the number of joins and the feature selection becomes less effective due to multiple comparison problems [4].

Figure 2 shows learning curves for classification accuracy,

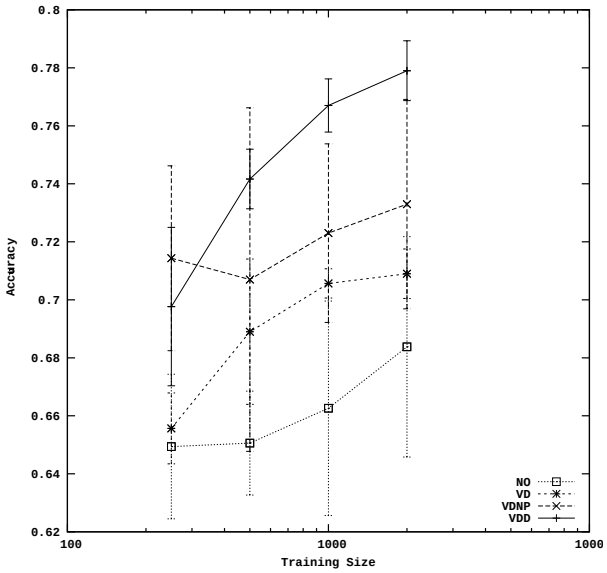


Figure 2: Learning curves: accuracy as a function of training-set size for NO, VD, VDNP, and VDD

including error bars of $\pm$ one standard deviation for the experiments exploring 12 joins. The learning curves show that increasing the training-set size always improves the generalization performance. The graph also highlights the different performance levels of the 4 levels of aggregation complexity. The more complex the aggregation, the better the performance. In addition, the most complex aggregation (VDD) has the smallest variance of the four contrasted methods.

Analysing the tree learned by C4.5 for the most complex model MVDD identifies the following variables as predictive: whether one underwriter was 'Hambrecht', the difference between the edit distances to the positive and negative reference vector of the underwiting banks, the number of IPO's previously underwritten by the head bank, the date of the IPO, the difference between edit distance of the head bank, the industry code of level 2 (2 digits), and the difference in Mahalanobis distance to the IPOs (Ticker) previously performed

Size	NO	AH	FOIL	Tilde	Lime	Progol	AC	LF
250	0.649	0.641	0.645	0.646	0.568	0.594	0.73	0.592
500	0.650	0.665	0.664	0.628	0.563	0.558	0.719	0.643
1000	0.662	0.701	0.658	0.630	0.530	0.530	0.724	0.638
2000	0.681	0.711	0.671	0.650	0.512	0.541	0.753	0.641

Table 5: Classification accuracy of methods independent of join depth

by the underwriting banks. This confirms our conclusions that the differences between the vector distances indeed play an important role for this task.

Table 5 contrasts the results for the other methods that are independent of the number of joins: abstraction hierarchies (AH) in the domain tables IND2 and IND, the four ILP systems FOIL, Tilde, Lime, and Progol, relational autocorrelation (AC), and logic-based feature construction (LF).

Including the values of the abstraction hierarchy (AH) improves slightly over no relational background knowledge, (suggesting that the industry classes are linked to exchange) but cannot compete with target-based set aggregation. The AC results show that there is significant degree of autocorrelation in this domain. Banks seem to operate primarily on one exchange or the other. AC outperforms all methods in this table, and only falls short of the best set-aggregation method MVDD in Table 4.

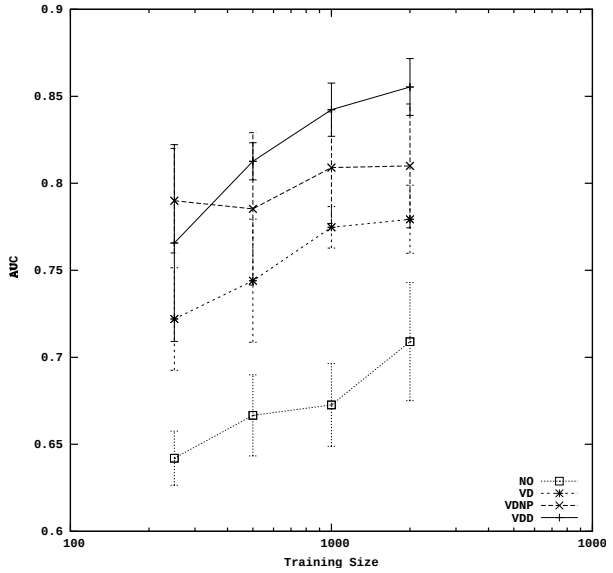


Figure 3: Learning curves: area under ROC as a function of training-set size for NO, VD, VDNP, and VDD

The two ILP systems FOIL and Tilde are still competitive for small data sets but for larger training sets fall short of using no relational background knowledge. There are three potential reasons for the low performance of the logic-based methods: (1) the task is noisy and the search mechanism within the system is overly sensitive to noise; (2) ILP systems are not optimized for numeric values, and/or (3) the relational domain properties (e.g. cardinality of the relationships) are not suitable for the particular systems. Logic-based systems can be used on simple feature-vector domains and have (on those domains) the same expressive power as a decision tree or a rule learner. However doing worse than C4.5 on the mostly numerical feature vectors suggests that the search strategy itself is not optimal for this task or that the regularization mechanism is insufficient and the systems overfit.

The low performance of LF is caused entirely by overfitting the training data since it contains, in addition to the binary features, all the original attributes from the target table IPO used by NO. The binary features are learned from the training set by optimizing classification performance. They are therefore very predictive on the training set and the decision tree overestimates their predictive performance.

The results for probability estimation (reported in Table 6) are similar to the results on accuracy. The most complex aggregation methods (MVDD or VDD) outperform the other methods and the performances increase in training size. Figure 3 shows the learning curves of NO,VD,VDPN, and VDD including error bars of $\pm$ one standard deviation for 12 joins.

Figure 4 shows the ROC curves for NO, MVD, MVDNP, and MVDD exploring 12 joins. MVDD and MVDNP present an interesting case where the ROC curves are crossing. MVDNP is better for high thresholds whereas MVDD performs better for lower thresholds. Analysing the probability

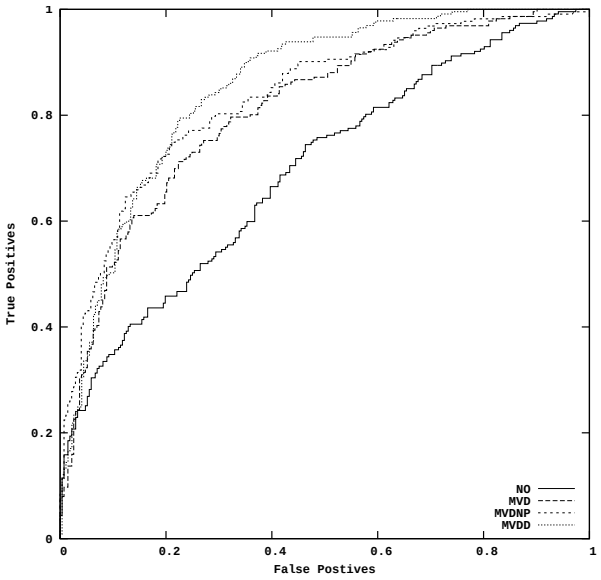


Figure 4: ROC curves for NO, MVD, MVDNP and MVDD

estimation performances of methods that are independent of

Size	NO	MOC	VD	MVD	MPN	VDPN	MVDPN	MD	VDD	MVDD
250: 6	0.642	0.697	0.717	0.691	0.672	0.748	0.716	0.68	0.729	0.734
250: 9	0.642	0.707	0.711	0.74	0.725	0.756	0.761	0.749	0.75	**0.764**
250:12	0.642	0.729	0.722	*0.755*	0.715	*0.79*	0.74	0.713	0.763	0.760
500: 6	0.666	0.702	0.738	0.741	0.72	0.746	0.739	0.75	0.774	0.79
500: 9	0.666	0.775	0.753	0.757	0.758	0.77	*0.802*	0.796	0.775	**0.821**
500:12	0.666	0.741	0.744	*0.787*	0.775	0.785	0.76	0.792	0.812	0.812
1000: 6	0.672	0.743	0.754	0.749	0.735	0.793	0.797	0.767	0.788	0.802
1000: 9	0.672	0.765	0.768	0.763	0.787	0.808	*0.825*	0.797	0.818	0.826
1000:12	0.672	0.778	0.774	*0.781*	0.78	0.809	0.797	0.793	**0.842**	0.829
2000: 6	0.709	0.727	0.744	0.752	0.732	0.795	0.796	0.787	0.794	0.824
2000: 9	0.709	0.785	0.772	0.781	0.807	0.805	*0.835*	0.799	0.832	0.838
2000:12	0.709	0.791	0.779	*0.801*	0.790	0.81	0.788	0.798	**0.855**	0.836

Table 6: Probability estimation performance (area under the ROC) for set aggregation methods, grouped by complexity

join depth in table 4.3 shows that the autocorrelation aggregator (AC) performs very well and almost reaches the performance of MVDD. Abstraction hierarchies (AH) are not as useful for probability estimation as they were for classification. Note, that ILP systems only predict a class label and therefore do not appear in the table.

Size	No	AH	AC	LF
250	0.642	0.63	0.79	0.626
500	0.666	0.673	0.814	0.694
1000	0.672	0.699	0.821	0.703
2000	0.709	0.714	0.838	0.702

Table 7: Probability estimation using AUC of methods independent of join depth

5 Conclusion and Future Work

We have shown that looking carefully at aggregation for relational learning creates a considerable design space for relational feature construction (either separately from learning or internally to a learning program). We are not aware of any learning program that considers even a small fraction of these aggregation operators, nor any that uses the more successful, target-dependent set aggregations.

Within the scope of the IPO domain the empirical results demonstrate that aggregation operators of higher complexity can significantly improve the generalization performance of relational learners. The best methods (VDD, MVDD) use target-dependent vector-distance aggregators (that transform the relational task into a conventional feature-vector representation that allows the use of conventional learning methods). An advantage of this transformation-based approach is its general applicability to regression, classification, and probability estimation tasks.

Our results furthermore show that for the same level of performance, increased aggregation complexity can trade off exploration depth. This is an important point since the size of the space increases exponentially in the search depth if the relations have a one-to-n or m-to-n cardinality. Scalability of relational learning is still an important research topic in relational learning [3].

The primary contribution of this work is the first detailed examination of aggregation for relational learning. Along with search through the relationship graph, aggregation is a major component of any relational learning method. We have shown, with the concept hierarchy, that with respect to aggregation there are various classes of relational learning problems, and that problems with high aggregation complexity can be deceptively simple in description.

Although quite suggestive, the generalizability of our positive findings (in favor of the more complex aggregators) is limited due to the focus on one particular domain and the limited maximum training size of 2000. Future work includes extending these experiments to multiple domains with different relational characteristics.

The presented aggregation methods are certainly not complete. Our findings motivate further exploration of potential aggregation methods. In particular there is still an open issue of numeric multidimensional and multi-type aggregation. Another open issue is the joint optimization of aggregation and model estimation. (Rather than treating them separately, as we have done.)

More generally, this work highlights that existing approaches to relational classification can show major performance differences. The field of relational learning still needs to develop a better understanding of why certain methods outperform on certain domains.

Acknowledgments

We are grateful to Jeff Siminoff, Sofus Macskassy, and William Greene for valuable comments and discussions. This work is sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-01-2-585. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

References

[1] Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[2] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. In *Pattern Recognition*, volume 30(7), pages 1145–1159, 1997.

[3] J. F′urnkranz. Dimensionality reduction in ILP: A call to arms. In Raedt L. and Muggleton S., editors, *Proceedings of the IJCAI-97 Workshop on Frontiers of Inductive Logic Programming*, 1997.

[4] D. Jensen and P.R. Cohen. Multiple comparisons in induction algorithms. In *Machine Learning*, volume 38, pages 309–338, 2000.

[5] D. Jensen and J. Neville. Data mining in social networks. In *Dynamic Social Networks Modeling and Analysis*, 2002.

[6] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *19th International Conference on Machine Learning*, 2002.

[7] M. Kirsten, S. Wrobel, and T. Horvath. Distance based approaches to relational learning and clustering. In Dzeroski Lavrac, editor, *RDM*, pages 213–232. Springer Verlag, 200.

[8] A. Knobbe, M. De Haas, and A. Siebes. Propositionalisation and aggregates. In *LNAI*, volume 2168, pages 277–288, 2001.

[9] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *AAAI/IAAI*, pages 580–587, 1998.

[10] S. Kramer and P. Flach N. Lavrac. *Propositionalization Approaches to Relational Data Mining*, pages 262–291. Springer-Verlag, 2001.

[11] L. Dehaspe L. De Raedt, H. Blockeel and W. Van Laer. Three companions for data mining in first order logic. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 105–139. Springer-Verlag, 2001.

[12] P.C. Mahalanobis. On the generalized distance in satistics. In *Proc. Natl. Institute of Science of India*, volume 12, pages 49–55, 1936.

[13] E. McCreath. Induction in first order logic from noisy training examples and fixed example set size. In *PhD Thesis*, 1999.

[14] S.H. Muggleton. Cprogol4.4: a tutorial introduction. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 105–139. Springer-Verlag, 2001.

[15] S.H. Muggleton and L. DeRaedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19 & 20:629–680, May 1994.

[16] A. Popescul, L. H. Ungar, S. Lawrence, and D. M. Pennock. Structural logistic regression: Combining relational and statistical learning. In *Proceedings of the Workshop on Multi-Relational Data Mining (MRDM-2002)*, pages 130–141. University of Alberta, Edmonton, Canada, July 2002.

[17] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Los Altos, California, 1993.

[18] J.R. Quinlan and R.M. Cameron-Jones. Foil: A midterm report. In P. Brazdil, editor, *Proceedings of the 6th European Conference on Machine Learning*, volume 667, pages 3–20. Springer-Verlag, 1993.

[19] B. Pfahringer S. Kramer and C. Helma. Stochastic propositionalization of non-determinate background knowledge. In *International Workshop on Inductive Logic Programming*, pages 80–94, 1998.

[20] A. Srinivasan and R.D. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, pages 352–367. Stockholm University, Royal Institute of Technology, 1996.