Scalable Statistical Learning for Relation Prediction on Structured Data

Yi Huang

München 2020

Scalable Statistical Learning for Relation Prediction on Structured Data

Yi Huang

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Yi Huang
aus Xinjiang, China

München, den 18.03.2020

Erstgutachter: Prof. Dr. Volker Tresp

Zweitgutachter: Professor Ning Zhong PhD

Tag der mündlichen Prüfung: 15.06.2020

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Huang, Yi

Name, Vorname

München, 18.03.2020

Ort, Datum

Yi Huang

Unterschrift Doktorand/in

Formular 3.2

Contents

Abstract **xv**

Zusammenfassung **xvii**

Acknowledgment **xix**

1 Introduction **1**

1.1 Motivation 1

1.2 Relational Knowledge Bases 3

1.2.1 Semantic Web 3

1.2.2 Linked Open Data 5

1.2.3 Ontological Background Knowledge 5

1.3 Statistical Learning for Relation Prediction 7

1.3.1 Statistical Relational Learning 7

1.3.2 Relational Graphical Models 8

1.3.3 Inductive Logic Programming 13

1.3.4 Tensor Decompositions 15

1.3.5 Remarks 16

1.4 Structure of the Thesis 18

1.5 Contributions of the Thesis 19

2 Learning with the Statistical Unit Node Set (SUNS) **23**

2.1 The Approach 24

2.1.1 Definition of Statistical Setting 24

2.1.2 Random Variables 25

2.1.3 Non-Random Covariates 26

2.1.4 Formal Definition of Statistical Setting 27

2.1.5 Multivariate Prediction Model: Reduced-Rank Penalized Regression 29

2.1.6 Transduction and Induction 32
2.2 Empirical Study: Friendship Prediction 33
2.2.1 Data Set and Experimental Setup 33
2.2.2 Results 37
2.3 Remarks 41

3 Kernel SUNS **43**
3.1 The Nyström Approximation 44
3.2 Kernel SUNS 45
3.3 Experiments 47
3.3.1 Scalability 47
3.4 Remarks 48

4 R-Model **51**
4.1 Object-Oriented Sampling Assumption 51
4.2 Relation-Oriented Sampling Assumption 52
4.3 An Example Illustrating R-Model 54
4.3.1 A Social Network 54
4.3.2 Modeling User-Movie Events 55
4.3.3 Adding Last Movie Watched 56
4.3.4 Adding Time of the Event 57
4.4 Empirical Study 1 57
4.5 Empirical Study 2 60
4.5.1 Methodology 61
4.5.2 Results 61
4.6 Remarks 64

5 Applications **65**
5.1 Stream Reasoning for Semantic Social Media Analysis 66
5.1.1 Stream Reasoning 66
5.1.2 Empirical Study 68
5.1.3 Evaluation 72
5.2 Life Science: Disease-Gene Prioritization 74
5.2.1 The Problem 74
5.2.2 Empirical Study 75
5.3 Location-based Personalized Recommendation 81

5.3.1 The BOTTARI Mobile Application 82
5.3.2 Data and Ontology 83
5.3.3 Empirical Study 87

6 Conclusions and Future Work **93**

Appendices **97**

List of Figures

1.1 Example of an RDF graph 4

2.1 Sketch of statistical unit node set 28
2.2 Entity-relationship diagram of the LJ-FOAF domain 33
2.3 Evaluated sampling strategies 35
2.4 Comparison between different algorithms for settings 1, 2, 5, 6 37
2.5 Comparison between different algorithms for settings 3, 4, 7, 8 39

3.1 The time complexity of the kernel model 49

4.1 Object-oriented vs. relation-oriented sampling assumption 53
4.2 A graphical model for the dependencies between users U and movies M. . 56
4.3 Integrate additional information 57
4.4 Experimental results without regularization 58
4.5 Experimental results with regularization 58
4.6 Experimental results by averaging individual models 60
4.7 HitRatio values 62
4.8 The nDCG scores for different factorization dimensions 63

5.1 Architecture of the stream reasoning 69
5.2 Entity relationship diagram of the social media data 70
5.3 Accuracy of top-N movie recommendations 73
5.4 Bio2RDF databases and connections 77
5.5 Genes vs. diseases as statistical units 79
5.6 Averaging the gene and disease models 80
5.7 Dataset statistics 84
5.8 Ontology modelling of the data. 86
5.9 Evaluation in Setting 1 89

5.10 Evaluation in Setting 2 90

List of Tables

2.1 Data statistics in the different experimental settings 38
2.2 NDCG scores averaged cross over the samples 38

5.1 Statistics of the data set 85
5.2 Number of ratings in different time frames 85
5.3 Comparisons with Trip Advisor and Rough Guide 91

Abstract

Relation prediction seeks to predict unknown but potentially true relations by revealing missing relations in available data, by predicting future events based on historical data, and by making predicted relations retrievable by query. The approach developed in this thesis can be used for a wide variety of purposes, including to predict likely new friends on social networks, attractive points of interest for an individual visiting an unfamiliar city, and associations between genes and particular diseases. In recent years, relation prediction has attracted significant interest in both research and application domains, partially due to the increasing volume of published structured data and background knowledge. In the Linked Open Data initiative of the Semantic Web, for instance, entities are uniquely identified such that the published information can be integrated into applications and services, and the rapid increase in the availability of such structured data creates excellent opportunities as well as challenges for relation prediction.

This thesis focuses on the prediction of potential relations by exploiting regularities in data using statistical relational learning algorithms and applying these methods to relational knowledge bases, in particular in Linked Open Data in particular. We review representative statistical relational learning approaches, e.g., Inductive Logic Programming and Probabilistic Relational Models. While logic-based reasoning can infer and include new relations via deduction by using ontologies, machine learning can be exploited to predict new relations (with some degree of certainty) via induction, purely based on the data. Because the application of machine learning approaches to relation prediction usually requires handling large datasets, we also discuss the scalability of machine learning as a solution to relation prediction, as well as the significant challenge posed by incomplete relational data (such as social network data, which is often much more extensive for some users than others).

The main contribution of this thesis is to develop a learning framework called the Statistical Unit Node Set (SUNS) and to propose a multivariate prediction approach used in the framework. We argue that multivariate prediction approaches are most suitable for

dealing with large, sparse data matrices. According to the characteristics and intended application of the data, the approach can be extended in different ways. We discuss and test two extensions of the approach–kernelization and a probabilistic method of handling complex n-ary relationships–in empirical studies based on real-world data sets. Additionally, this thesis contributes to the field of relation prediction by applying the SUNS framework to various domains. We focus on three applications:

1. In social network analysis, we present a combined approach of inductive and deductive reasoning for recommending movies to users.

2. In the life sciences, we address the disease gene prioritization problem.

3. In the recommendation system, we describe and investigate the back-end of a mobile app called BOTTARI, which provides personalized location-based recommendations of restaurants.

Zusammenfassung

Die Beziehungsvorhersage strebt an, unbekannte aber potenziell wahre Beziehungen vorherzusagen, indem fehlende Relationen in verfügbaren Daten aufgedeckt, zukünftige Ereignisse auf der Grundlage historischer Daten prognostiziert und vorhergesagte Relationen durch Anfragen abrufbar gemacht werden. Der in dieser Arbeit entwickelte Ansatz lässt sich für eine Vielzahl von Zwecken einschließlich der Vorhersage wahrscheinlicher neuer Freunde in sozialen Netzen, der Empfehlung attraktiver Sehenswürdigkeiten für Touristen in fremden Städten und der Priorisierung möglicher Assoziationen zwischen Genen und bestimmten Krankheiten, verwenden. In den letzten Jahren hat die Beziehungsvorhersage sowohl in Forschungs- als auch in Anwendungsbereichen eine enorme Aufmerksamkeit erregt, aufgrund des Zuwachses veröffentlichter strukturierter Daten und von Hintergrundwissen. In der Linked Open Data-Initiative des Semantischen Web werden beispielsweise Entitäten eindeutig identifiziert, sodass die veröffentlichten Informationen in Anwendungen und Dienste integriert werden können. Diese rapide Erhöhung der Verfügbarkeit strukturierter Daten bietet hervorragende Gelegenheiten sowie Herausforderungen für die Beziehungsvorhersage.

Diese Arbeit fokussiert sich auf die Vorhersage potenzieller Beziehungen durch Ausnutzung von Regelmäßigkeiten in Daten unter der Verwendung statistischer relationaler Lernalgorithmen und durch Einsatz dieser Methoden in relationale Wissensbasen, insbesondere in den Linked Open Daten. Wir geben einen Überblick über repräsentative statistische relationale Lernansätze, z.B. die Induktive Logikprogrammierung und Probabilistische Relationale Modelle. Während das logikbasierte Reasoning neue Beziehungen unter der Nutzung von Ontologien ableiten und diese einbeziehen kann, kann maschinelles Lernen neue Beziehungen (mit gewisser Wahrscheinlichkeit) durch Induktion ausschließlich auf der Basis der vorliegenden Daten vorhersagen. Da die Verarbeitung von massiven Datenmengen in der Regel erforderlich ist, wenn maschinelle Lernmethoden in die Beziehungsvorhersage eingesetzt werden, diskutieren wir auch die Skalierbarkeit des maschinellen Lernens sowie die erhebliche Herausforderung, die sich aus unvollständigen relationalen Daten ergibt

(z. B. Daten aus sozialen Netzen, die oft für manche Benutzer wesentlich umfangreicher sind als für Anderen).

Der Hauptbeitrag der vorliegenden Arbeit besteht darin, ein Lernframework namens Statistical Unit Node Set (SUNS) zu entwickeln und einen im Framework angewendeten multivariaten Prädiktionsansatz einzubringen. Wir argumentieren, dass multivariate Vorhersageansätze am besten für die Bearbeitung von großen und dünnbesetzten Datenmatrizen geeignet sind. Je nach den Eigenschaften und der beabsichtigten Anwendung der Daten kann der Ansatz auf verschiedene Weise erweitert werden. In empirischen Studien werden zwei Erweiterungen des Ansatzes–ein kernelisierter Ansatz sowie ein probabilistischer Ansatz zur Behandlung komplexer n-stelliger Beziehungen– diskutiert und auf realen Datensätzen untersucht. Ein weiterer Beitrag dieser Arbeit ist die Anwendung des SUNS Frameworks auf verschiedene Bereiche. Wir konzentrieren uns auf drei Anwendungen:

1. In der Analyse sozialer Netze stellen wir einen kombinierten Ansatz von induktivem und deduktivem Reasoning vor, um Benutzern Filme zu empfehlen.

2. In den Biowissenschaften befassen wir uns mit dem Problem der Priorisierung von Krankheitsgenen.

3. In den Empfehlungssystemen beschreiben und untersuchen wir das Backend einer mobilen App “BOTTARI”, das personalisierte ortsbezogene Empfehlungen von Restaurants bietet.

Acknowledgment

First and foremost, I owe my deepest gratitude to Prof. Dr. Volker Tresp, who has played a particularly significant role in my intellectual development. Prof. Dr. Tresp introduced me to the field of statistical machine learning, and his enthusiasm, incisive thoughts, open-mindedness, and humor made my research a memorable and joyful journey. One could not wish for a better or friendlier supervisor.

I am especially thankful to Prof. Zhong Ning as well, who kindly agreed to allocate some of his time to supervising my thesis despite the significant demands of his research and teaching work. I would also like to thank Prof. Dr. Andreas Butz for being the president of my promotion committee.

I would like to express my gratitude to Prof. Dr. Hans-Peter Kriegel. He has supported me throughout my thesis with his patience, encouragement, and constructive suggestions. He supervised my Diploma thesis and introduced me to the Machine Learning research group led by Prof. Dr. Volker Tresp at Siemens AG. I was impressed by his meticulous scholarship and academic guidance.

I am additionally grateful to Dr. Michael May, the leader of the Technology Field Business Analytics and Monitoring at Siemens AG, and Dr. Ulli Waltinger, the leader of the Machine Intelligence Research Group at Siemens AG, for their constant support to both my scientific research and my career plan.

This thesis would not have been possible without the support of many other individuals. I would like to thank, including Dr. Kai Yu, Dr. Shipeng Yu, Prof. Dr. Achim Rettinger, Dr. Markus Bundschus, Dr. Maximilian Nickel, PhD Emanuele Della Valle, Irene Celino, Hendrik Wermser, and Dr. Xueyan Jiang.

My special thanks go to the staff of the Institute of Informatics of the Ludwig Maximilian University of Munich and my colleagues at Siemens AG (Corporate Technology, Munich) who provided a welcoming environment in which to pursue my work. I also acknowledge funding by the German Federal Ministry of Economy and Technology (BMWi) under the THESEUS project and by the EU FP 7 under the Integrated Project LarKC.

Finally, I am grateful to my parents and my wife for their patience and love. Without them, this work never would have begun.

Chapter 1

Introduction

1.1 Motivation

The Semantic Web structures, annotates, and renders the World Wide Web. A remarkable deal of development has occurred around the Linked Open Data (LOD) initiative, where the term Linked Data describes a method of exposing, sharing, and connecting data via dereferenceable Unique Resource Identifiers (URIs) on the Web. Typically, existing structured and unstructured data sources are published in the Semantic Web in the form of Resource Description Frameworks (RDFs), where statements are expressed as simple subject-property-object (s, p, o) triples and are graphically displayed as a directed labeled link between a node representing the subject and a node representing the object (Figure 1.1). In the LOD cloud, data sources are interlinked with other data sources. In recent years, the Knowledge Graph has rapidly developed in this field and plays an increasing important role in industrial applications.

In some research work and applications, subsets of the LOD cloud are retrieved from repositories, and some form of logical reasoning is applied to materialize implicit triples. The number of inferred triples is typically on the order of the number of explicit triples, but it is assumed that there exist a vast number of additional real triples that are neither known as facts nor able to be derived by logical reasoning. This assumption concerns both triples within contributing data sources, such as DBpedia[1], and triples describing interlinks between contributing data sources.

The goal of the thesis is to predict relations in structured datasets by using statistical machine learning approaches and exploiting regularities of the datasets. In order to do so, we must take into account the nature of the data in the LOD cloud, which has been

[1] http://dbpedia.org/

dynamically evolving and is now quite noisy. As a result, an approach that focuses on flexibility and ease of use is preferable to highly sophisticated approaches that can only be applied by machine learning experts. Reasonable requirements for such approaches are as follows.

- The use of machine learning approaches should require a minimum of user intervention.
- Learning algorithms should scale well with the size of the Semantic Web.
- The relations (triples) and their probabilities predicted by machine learning approaches should be easily queried.[2]
- Learning approaches should be suitable to the characteristics of the LOD data: high sparsity (e.g., only small numbers of users who are friends in a social network) and missing information (e.g., certain users chose not to reveal private information).

Many algorithms have been proposed in the past for relation prediction in the Semantic Web, including several based on recent work in statistical relational learning (see [96] for an overview). One family of approaches apply a global probabilistic model to a segment of a Semantic Web knowledge-base and is able to predict the probability of statements in the domain (examples are [33, 27, 106, 52]). In these approaches, the states of probabilistic nodes in a graphical model represent the truth values of statements. Although these approaches are attractive, the sheer size of the Semantic Web and the vast number of potentially true statements it contains reduce the utility of such approaches in many large-scale applications. The second family of approaches consists of conditional models that define a classification problem and attempt to derive appropriate relational features that can be used by predicting a target class. These approaches include Inductive Logic Programming (ILP) [83, 75] and propositionalized ILP (pILP) approaches [23, 64]. Since the size of the data samples can be controlled, ILP and pILP easily achieve scalability, but conditional models have problems with missing data. The approach to relation prediction that we propose in this thesis aims to combine the advantages of both of these families of approaches and meet all above mentioned requirements.

[2]SPARQL is a standard for querying RDF-specific information and for displaying query results.

1.2 Relational Knowledge Bases

Data can be categorized into two classes: structured data, e.g., relational databases, and unstructured data, e.g., texts and images.[3] Traditionally, structured data is stored, processed, and analyzed in files or database systems. For instance, Knowledge Discovery in Databases (KDD) analyzes database systems in order to discern trends, patterns, and rules in the data they contain. Our machine learning approaches for relation prediction apply to structured data as well—however, unlike KDD, our approach addresses structured and semantically annotated data published on the Web rather than in databases.

This section describes the standard data model in the Semantic Web (the Resource Description Framework), presents SPARQL Protocol and RDF Query Language (SPARQL), an RDF query language containing aggregation functions that are needed by our approach, briefly introduces the Linked Open Data we use in empirical studies in this thesis as well as the real applications in Chapter 5, and explains Web Ontology Language (OWL), Rule Markup Language and the reasoning tasks our approach performs on the relational knowledge bases the Semantic Web contains.

1.2.1 Semantic Web

RDF: A Data Model for the SW

The recommended data model for the Semantic Web (SW) is the *Resource Description Framework (RDF)*. The RDF was developed to represent information about resources on the WWW (e.g., metadata and annotations) where a resource is understood to be a thing that can be uniquely identified via a uniform resource identifier (URI). The basic statement expressing a fact observed is a triple of the form *(subject, property, property value)* or, equivalently, *(subject, predicate, object)*. A triple can be described graphically as a directed arc, labeled by the property (predicate) and pointing from the subject node to the property value node. A complete database (triple store) can then be displayed as a directed graph (see the example in Figure 1.1).

RDF Schema (RDFS) and various dialects of OWL (Ontology Web Language) can be used to encode semantic constraints. Concepts and simple relationships between concepts are defined in RDFS, while OWL ontologies build on RDFs/RDFSs and improve expressiveness. More details on SW standards can be found in [2, 40].

[3] Besides structured data and unstructured data, some data is semi-structured such as XML (eXtensible Markup Language) or weakly structured. Those types of data are outside of the scope of this thesis.

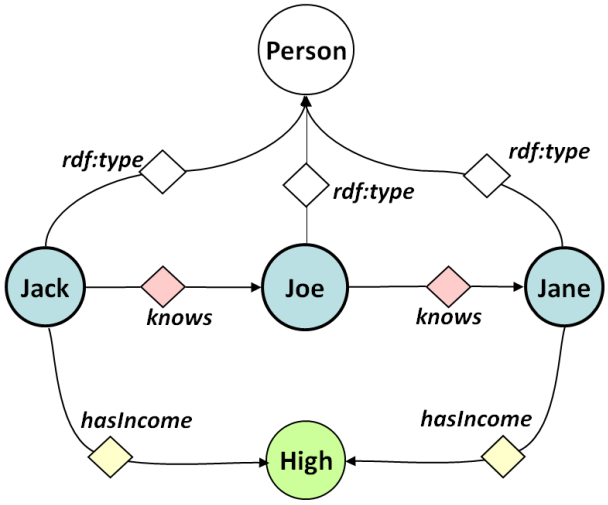


Figure 1.1: Example of an RDF graph displaying a social friendship network in which the income of a person is an attribute. Circular nodes and triples represent resources, and triples are represented by labeled directed links from a subject node to an object node. The diamond-shaped nodes stand for random variables that are in state *one* if the corresponding triples exist. Nodes representing statistical units (here: *Persons*) have a darker rim.

The Query Language SPARQL

SPARQL is a standard language for querying RDF-specific information and for displaying the results. A SPARQL query searches for subgraphs as its primary function, but it can also formulate more expressive query patterns, apply filters, and format the output. A SPARQL query must contain a `PREFIX` statement for specifying the namespace, a `SELECT` statement that determines the output pattern (typically a table of variable bindings) and a `WHERE` statement that specifies the searchable graph pattern and that might contain variables. More complex queries are possible via grouping, optional patterns, and alternative patterns. Filters can be used to restrict the search pattern further. Filters might include numerical comparisons ($<, >, =$), special operators (e.g., for strings and data time), Boolean operators, and arithmetic operations. The output format can be modified via `CONSTRUCT`, `DESCRIBE` and `ASK`. With `CONSTRUCT` the output can be formatted as an RDF document. `MODIFY` can be used to manipulate the output pattern. The keywords `ORDER BY`, `DISTINCT` can be used to reduce redundancy in the result set. Importantly, SPARQL inherently supports aggregation functions such as count, sum, avg, max, and min. (Details on https://www.w3.org/TR/rdf-sparql-query/)

1.2.2 Linked Open Data

The Linked Open Data (LOD) initiative is the most significant component of the Semantic Web. The term *Linked Data* describes a method of exposing, sharing, and connecting data via dereferenceable Unique Resource Identifiers (URIs) on the Web. Typically, existing structured and unstructured data sources are published to the Semantic Web in the form of RDF, and data sources are interlinked with other data sources in the LOD cloud (https://lod-cloud.net/). Subsets of the LOD cloud are retrieved into repositories, and some form of logical reasoning is applied to materialize implicit triples (cf. Section 1.2.3).

1.2.3 Ontological Background Knowledge

Ontologies build on RDFs/RDFSs and add expressiveness. W3C developed standards for the Web Ontology Language (OWL), which comes in three dialects or profiles, the most expressive of which is OWL Full, which is a true superset of RDFS. A full inference procedure for OWL Full is not implementable with simple rule engines [38]. Some applications requiring OWL Full might build an application-specific reasoner instead of using a general one. OWL Full includes OWL Description Language (OWL DL), and OWL DL includes OWL Lite. Both OWL DL and OWL Lite are decidable but are not true supersets of RDFS.

In OWL, one can state that classes are equivalent or disjointed, and instances of which are identical or different, respectively. The behavior of properties can be classified into different types, such as symmetric, transitive, functional, or inverse functional (e.g., teaches is the inverse of isTaughtby). In RDFS, concepts are named, while OWL allows the user to construct classes by enumerating their content (explicitly stating its members), through forming intersections, unions, and complements of classes. Also, classes can be defined via property restrictions. For example, the constraints that (1) first-year courses must be taught by professors, (2) mathematics courses are taught by David Billington, (3) all academic staff members must at least teach one undergraduate course, can all be expressed in OWL using the constructs `owl:allValuesFrom` ($\forall$), `owl:hasValue`, and `owl:someValuesfrom` ($\exists$). Furthermore, cardinality constraints can be formulated by using `owl:maxCardinality` and `owl:minCardinality`: for instance, a course must be taught by someone; a department must have at least ten and at most 30 members (Examples from [2]). The fact that both instances and ontologies can be joined by simply joining the corresponding graphs makes OWL very attractive: the only real thing is the (RDF-)graph [38].

In some semantic-poor applications, ontologies have only relevance in terms of the

definition of classes and properties. Conversely, in some semantic-rich domains, such as bioinformatics, medical informatics and some industrial applications [80], sophisticated ontologies have already been developed [38].

Reasoning

An ontology formulates logical statements that can be used for analyzing data consistency and for deriving new implicit statements concerning instances and concepts. Including implicit statements into SPARQL makes querying more powerful. Inferred closure consists of the extension of a knowledge base on the Semantic Web, along with all the implicit statements, that could be inferred from it, using enforced semantics [55]. In the *materialization* strategy, after each update to the knowledge base, the repository assures that the inferred closure is computed, updated, and made available for query evaluation or retrieval. As a reasoning strategy, total materialization is utilized by several popular Semantic Web repositories, including some of the standard configurations of Sesame and Jena (https://jena.apache.org/). Total materialization denotes the calculation of all implicit triples at loading time, which might be preferred if query response time is critical [55]. Note, that total materialization is only feasible in some restricted ontologies.

In this thesis, we consider relation prediction as probabilistic materialization, i.e., the materialization of statements weighted by their estimated probabilities. In the following, we will assume that *logical* materialization has been performed before *probabilistic* materialization such that the statements that can logically be inferred are available for relation prediction.

Rules

RuleML (Rule Markup Language) is a rule language formulated in XML and based on datalog, a function-free fragment of Horn clausal logic. RuleML allows the formulation of if-then-type rules. Both RuleML and OWL DL are different subsets of first-order logic (FOL). SWRL (Semantic Web Rule Language)[4] is a proposal for a Semantic Web rule language combining sublanguages of OWL (OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog). Datalog clauses are essential for modeling background knowledge in cases where DL might be inappropriate in many industrial applications.

[4]https://www.w3.org/Submission/SWRL/

1.3 Statistical Learning for Relation Prediction

1.3.1 Statistical Relational Learning

Global Probabilistic Models

There are several approaches for learning in relational domains, including approaches in which a global probabilistic model in the form of a probabilistic graphical model is learned, e.g., [33, 27, 106, 52]. The state of a probabilistic node in these models corresponds to the truth value of the corresponding atomic statement.[5] Formally, let $X^{(s,p,o)} = 1$ mean the statement (s, p, o) is true; otherwise, let $X^{(s,p,o)} = 0$. The most natural quantity that could be defined as a statement probability is

$$P(X^{(s,p,o)} = 1|\text{KB}),$$

which is the marginal probability of $X^{(s,p,o)}$ given the information in the Semantic Web knowledge base (KB). The probability can be decomposed as

$$P(X^{(s,p,o)}|\text{KB}) = \sum_{\{X^U\}} P(X^{(s,p,o)}, \{X^U\}|\text{KB})$$

where $\{X^U\}$ stands for the set of *all* statements whose truth values are unknown. Certainly, simplifications can be applied such that this sum can be calculated (approximately) for relatively large networks (e.g., [27]), but it needs to be shown that Web-size scalability is feasible. A great advantage of this approach is that it is resilient when faced with missing information, meaning that it can handle arbitrary patterns of missing information.

Conditional Models

A second family of approaches includes the traditional approaches from ILP [83, 75, 58, 23, 64] and a number of related statistical approaches [96, 82]. Typically, in such approaches, a classification problem is stated. For example, the task might be to assign an entity to an ontological class or to predict a particular property of an entity (e.g., the high income of a person). Here we assume that the target class corresponds to a node $X^{(s,p,o)}$. Learning consists of generating relational features that are good predictors for the target class. For example, one might be able to predict income from the number of rooms in a person's house or the income of the person's friends. The relational features are calculated from

[5] A probabilistic node is simply the graphical representation of a random variable, representing in our case the truth value of a basic statement or triple. Not to be confused with a node in an RDF-graph.

nodes in the neighborhood of the entity of interest. The nodes that render the target class independent of the remaining probabilistic nodes form the Markov blanket $\mathrm{MB}^{(s,p,o)}$ such that

$$P(X^{(s,p,o)})|\mathrm{KB}) \approx P(X^{(s,p,o)}|\mathrm{MB}^{(s,p,o)}).$$

In the situations we are considering, this approach is difficult to apply due to a large number of statements with unknown truth values. ILP solves the problem of unknown truth values by merely making a closed-world assumption (thus there are no missing truth values in the Markov blanket), which is not appropriate in the context of the SW.[6] Due to the closed-world assumption, data points derived from Markov blanket models are independent, and the number of instances in the training set is under the control of the user, thus guaranteeing scalability.

Discussion

In conclusion, a global model can more easily deal with missing information but might not scale well, whereas conditional models scale better but have problems with missing information. Thus we pursue a model that attempts to combine the advantages of both approaches by being able to handle missing data and by being scalable. In addition, the approach should be able to deal well with sparse data. In the next chapter, we define a general statistical setting for learning approaches, including the statistical unit, population, and sample. We discuss suitable algorithms for this setting and propose a model based on the statistical unit node set. Using such models, one can predict potential relations not only within the sample (through transduction) but also for statistical units outside the sample in the population (via induction). The learned probabilistic relations can be stored subsequently in the Semantic Web knowledge-base as weighted triples through certain techniques, e.g., reification.

1.3.2 Relational Graphical Models

The probability distribution in a directed RGM, i.e., a relational Bayesian model, can be written as

$$P(\vec{U} = \vec{u}) = \prod_{U \in \vec{U}} P(U|par(U)).$$

U is represented as a node in a Bayesian network and arcs point from all parent nodes $par(U)$ to the node U. One partitions all elements of $\vec{U}$ into node-classes, and each U

[6] A discussion on open-world and closed-world reasoning for the SW can be found in [34].

belongs to precisely one node-class. The key property of all U in the same node-class is that their local distributions are identical, which means that $P(U|par(U))$ is the same for all nodes within a node-class and can be described by a truth-table or more complex representations such as decision trees. For example, all nodes representing the IQ scores of students in a university might form a node class, all nodes representing the difficulties of university courses might form another node-class, while the nodes representing the grades of students in courses might form a third.

Probabilistic Relational Models

PRMs were one of the first published RGMs and attracted the interest of the statistical machine learning community [57, 33]. PRMs combine a frame-based logical representation with probabilistic semantics based on directed graphical models. The nodes in a PRM model the probability distribution of object attributes, whereas the relationships between objects are assumed to be known. Naturally, this assumption simplifies the model substantially. In the context of the SW, object attributes would primarily correspond to object-to-literal statements. In subsequent papers, PRMs have been extended to consider cases where the relationships between objects (in the context of the SW, these would roughly be object-to-object statements) are unknown, which involve a type of uncertainty referred to as *structural uncertainty* in the PRM framework [33]. More straightforward cases, where one of the objects in a statement is known, while the partner object is unknown, display *reference uncertainty*. In cases of reference uncertainty, the number of potentially true statements is assumed to be known, which means that only an equivalent number of random nodes need to be introduced. The second form of structural uncertainty found in the PRM framework is *existence uncertainty*, where binary random variables are introduced, representing the truth values of relationships between objects.

For some PRMs, regularities in the PRM structure can be exploited (encapsulation), and exact inference is possible. Large PRMs require approximate inference through methods such as loopy belief propagation. Learning in PRMs is likelihood-based or based on empirical Bayesian learning. Structural learning typically uses a greedy search strategy, where one needs to guarantee that the ground Bayesian network does not contain directed loops.

More Directed RGMs

A *Bayesian logic program* is defined as a set of Bayesian clauses [53]. A Bayesian clause specifies the conditional probability distribution of a random variable given its parents on a template level, i.e., in a node-class. A special feature of Bayesian logic programs is that, for

a given random variable, *several* such conditional probability distributions might be given. As an example, *bt(X) | mc(X)* and *bt(X) | pc(X)* specify the probability distribution for blood type given the two different dispositions *mc(X)* and *pc(X)*. The truth value for *bt(X) | mc(X), pc(X)* can then be calculated based on various combination rules (e.g., noisy-or). In a Bayesian logic program, for each clause, there is one conditional probability distribution and for each Bayesian predicate (i.e., node-class) there is one combination rule. *Relational Bayesian networks* [47] are related to Bayesian logic programs and use probability formulae for specifying conditional probabilities. *Relational dependency networks* [78] also belong to the family of directed RGMs and use decision trees to learn the dependency of a node given its Markov blanket.

Markov Logic Networks

The probability distribution of an undirected graphical model or Markov network can be written as

$$P(\vec{U} = \vec{u}) = \frac{1}{Z} \prod_k g_k(u_k)$$

where $g_k(.)$ is a potential function, u_k is the state of the k-th clique and Z is the partition function normalizing the distribution. A more convenient log-linear representation of the form

$$P(\vec{U} = \vec{u}) = \frac{1}{Z} \exp \sum_k w_k f_k(u_k)$$

is often preferred, where the feature functions f_k can be any real-valued function and where $w_i \in \mathbb{R}$.

We will discuss two major approaches that use this representation: Markov logic networks and relational Markov models.

Let F_i be a formula in first-order logic and let $w_i \in \mathbb{R}$ be a weight attached to each formula. Then a MLN L is defined as a set of pairs (F_i, w_i) [89] [27]. One introduces a binary node for each possible grounding of each predicate appearing in L (in the context of the SW, we would introduce a node for each possible statement), given a set of constants $c_1, \ldots, c_{|C|}$. The state of the node is equal to 1 if the ground atom/statement is true, and 0 otherwise (for an N-ary predicate, there are $|C|^N$ such nodes). A grounding of a formula is an assignment of constants to the variables in the formula (considering formulas that are universally quantified). If a formula contains N variables, then there are $|C|^N$ such assignments. The nodes in the Markov network $M_{L,C}$ are the grounded predicates. Besides, the MLN contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. w_i is the weight

associated with F_i in L. A Markov network $M_{L,C}$ is a grounded Markov logic network of L with

$$P(\vec{U} = \vec{u}) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(\vec{u})\right)$$

where $n_i(\vec{u})$ is the number of formula groundings that are true for F_i. MLN makes the unique names assumption, the domain closure assumption and the known function assumption, but all of these assumptions can be relaxed.

An MLN puts weights on formulas: the larger the weight, the higher the confidence that the formula is true. When all weights are equal and become infinite, one enforces the formulas strictly, and all worlds that agree with the formulas have the same probability.

The simplest form of inference concerns the prediction of the truth value of a grounded predicate given the truth values of other grounded predicates (conjunction of predicates) for which the authors of [89] present an efficient algorithm. In the first phase, the minimal subset of the ground Markov network that is required to calculate the conditional probability is returned. This subset must be small since, in the worst case, an inference could involve all nodes. In the second phase, Gibbs sampling is used to perform inference on this reduced network.

Learning consists of estimating the w_i. In learning, MLN makes a closed-world assumption and employs a pseudo-likelihood cost function, which is the product of the probabilities of each node given its Markov blanket. Optimization is performed using a limited memory BFGS algorithm.

Finally, there is the issue of structural learning, which, in this context, defines the employed first-order formulae. Some formulae are typically defined by a domain expert *a priori*. Additional formulae can be learned by directly optimizing the pseudo-likelihood cost function or by using ILP algorithms. For the latter, in [89], the authors use CLAUDIAN [24], which can learn arbitrary first-order clauses (not just Horn clauses, as with many other ILP approaches).

Latent Class RGMs

The infinite hidden relational model (IHRM) [106] presented here is a directed RGM (i.e., a relational Bayesian model) with latent variables.[7] The IHRM is formed as follows. First, we partition all objects into classes $K_1, ...K_{|K|}$, using, for example, ontological class information. For each object in each class, we introduce a statement (*Object, hasHiddenState, H*). If *Object* belongs to class K_i, then $H \in \{1, \ldots, N_{K_i}\}$, i.e., the number of states of H is

[7] Kemp et al. [52] presented an almost identical model independently.

class-dependent. As before, we introduce a random variable or node U for each potentially true basic statement (grounded atom). Let Z_{Object} denote the random variables that involve *Object* and H. Z_{Object} is a latent variable or latent node since the true state of H is unknown. $Z_{Object} = j$ stands for the statement that $(Object,\ hasHiddenState, j)$.

We now define a Bayesian network where the nodes Z_{Object} have no parents, and the parents of the nodes for all other statements are the latent variables of the objects appearing in the statement. In other words, if U stands for the fact that $(Object_1, property, Object_2)$ is true, then there are arcs from Z_{Object_1} and Z_{Object_2} to U. The object-classes of the objects in a statement together with the property define a node-class for U. If the property value is a literal, then the only parent of U is Z_{Object_1}.

In the IHRM we allow the number of states in each latent node to be infinite and use the formalism of Dirichlet process mixture models. During inference, only a small number of the infinite states are occupied, leading to a clustering solution where the number of states in the latent variables N_{C_i} is automatically determined.

Since the dependency structure in the ground Bayesian network is local, it might seem that only local information influences prediction. This impression is incorrect, since, in the ground Bayesian network, common children U with evidence lead to interactions between the parent latent variables. Thus information can propagate in the network of latent variables. Training is based on various forms of Gibbs sampling (e.g., the Chinese restaurant process) or mean-field approximations. Training only needs to consider random variables U corresponding to statements that received evidence, e.g., statements that are either known to be true or known not to be true; random variables that correspond to statements with unknown truth value (i.e., without evidence) can be ignored entirely.

IHRM has several key advantages. First, no structural learning is required, since the directed arcs in the ground Bayesian network are directly given by the structure of the SW graph. Second, the IHRM model can be thought of as an infinite relational mixture model, realizing hierarchical Bayesian modeling. Third, the mixture model allows a cluster analysis providing insight into the relational domain.

The IHRM has been applied to recommender systems in medical domains and for gene function prediction, and it was the first relational model applied to trust learning [88]. In [86], the authors showed how ontological class information can be integrated into the IHRM.

1.3.3 Inductive Logic Programming

Inductive logic programming (ILP) encompasses various approaches that attempt to learn logical clauses. In light of the discussion in the last section, ILP uses logical (binary) features derived from logical expressions, typically conjunctions of (negated) atoms. Recent extensions on probabilistic ILP have also address uncertain domains.

ILP Overview

This section is on "strong" ILP, which covers the majority of ILP approaches and concerns classification of statistical units and on predicate definition[8]. Strong ILP performs modeling in relational domains that are somewhat related to the approach discussed in the previous section. Let us consider FOIL (First Order Inductive Learner) as a representative example [83]. The outcome of FOIL is a set of definite clauses (a particular if-then rule) with the same head (then-part).

Here is an example (modified from [29]). Let the statistical unit be a customer with ID *CID*. $VC = 1$ indicates that someone is a valuable customer, $GC = 1$ indicates that someone owns a golden credit card and $SB = 1$ indicates that someone would buy a sailboat. The first rule that FOIL might have learned is that a person is interested in buying a sailboat if this person owns a gold card. The second rule indicates that a person would buy a sailboat if this person is older than 30 and has at least once made a credit card purchase of more than 100 Euro:

$$\begin{aligned} \textit{sailBoat(CID, } SB = 1) &\leftarrow \textit{customer(CID,}GC = 1) \\ \textit{sailBoat(CID, } SB = 1) &\leftarrow \textit{customer(CID, Age)} \\ &\wedge \textit{purchase(CID, PID, Value, PM)} \\ &\wedge \textit{PM = credit-card} \wedge \textit{Value} > 100 \wedge \textit{Age} > 30. \end{aligned} \tag{1.1}$$

In rule learning, FOIL uses a covering paradigm, and derives the first rule to correctly predict as many positive examples as possible (covering) with a minimum number of false positives. Subsequent rules then try to cover the remaining positive examples. The head of a rule (then-part) is a predicate, and the body (the if-part) is a product of (negated) atoms containing constants and variables.[9] Naturally, there are many variants of FOIL. FOIL uses a top-down search strategy for refining rule bodies, whereas PROGOL [74] uses a bottom-up strategy and GOLEM [75] uses a combined strategy. Furthermore, FOIL

[8] A predicate definition is a set of program clauses with the same predicate symbol in their heads.

[9] FOIL learning is called learning from entailment in ILP terminology.

uses a conjunction of atoms and negated atoms in the body, whereas other approaches use PROLOG constructs. The community typically discusses these different approaches in terms of language bias (which rules the language can express), search bias (which rules can be found) and validation bias (when validation tells the user to stop refining a rule). An advantage of ILP is that non-grounded background knowledge also can be taken into account (typically in the form of a set of definite clauses that might be part of an ontology).

Given the discussion in the last section, the statistical unit corresponds to a customer, and FOIL introduces a binary target feature (1) for the target predicate *sailBoat(CID, SB)*. The second feature (2) is one if the customer owns a golden credit card and zero otherwise. Then a view is generated with attribute CID. A CID is entered in that view each time the person makes a credit card purchase of more then 100 Euro, but only if that person is more than 30 years old. The third feature (3) is binary and is equal to one if the CID is present in the view at least once and zero otherwise. FOIL then applies an elementary combination rule: if the feature (2) or feature (3) equal to one for a customer, then the target feature (1) is true.

Propositionalization, Upgrading, and Lifting

ILP approaches like FOIL can be decomposed into the generation of binary features (based on rule bodies) and a logical combination, which in the case of FOIL, is quite simple. As stated before, ILP approaches contain a sophisticated search strategy for defining optimal rule bodies. In contrast, if the generation of rule bodies is performed as a preprocessing step, the process is referred to as propositionalization [58]. Binary features are often collected through simple joins of all possible attributes. An early approach to propositionalization is LINUS [64]. Instead of using the simple FOIL combination rule, other feature-based learners are often used. Although it has been proven that propositionalization is inefficient in some special cases [23], propositionalization has produced excellent results.

The inverse process to propositionalization is called *upgrading* (or lifting) [102] and turns a propositional feature-based learner into an ILP learner. The main difference between upgrading and propositionalization is that the former optimizes features concerning the performance of the overall system. In practice, many strong ILP systems can be interpreted as upgraded propositional learners: FOIL is an upgrade of the propositional rule-induction program CN2, and PROGOL can be viewed as upgrading the AQ approach to rule induction. Additional upgraded systems include Inductive Classification Logic (ICL [25]) which uses classification rules, TILDE [13] and S-CART, which use classifica-

tion trees, and RIBL [31], which uses nearest neighbor classifiers. nFOIL [61] combines FOIL with a Naive Bayes (NB) classifier by changing the scoring function and by introducing probabilistic covering. nFoil was able to outperform FOIL and propositionalized NB on standard ILP problems. kFoil [60] is another variant that derives kernels from FOIL-based features.

Discussion

ILP algorithms can easily be applied to the SW if we identify atoms with basic statements. ILP fits well into the deterministic framework of the SW. In many ways, statistical SW learning is related to ILP's propositionalization; the main difference is the principled statistical framework of the former. Thus most of the discussion on scalability carries over to ILP's propositionalization. When ILP's complex search strategy for defining optimal rule bodies is applied, training time increases but remains proportional to the number of samples. An interesting new aspect of ILP is that it produces definite clauses that can be integrated–possibly with some restrictions–into the Semantic Web Rule Language. ILP approaches that consider learning with description logic (and clauses) are described, for example, in [20, 90, 67, 69, 70, 68]. An empirical study can be found in [30].

1.3.4 Tensor Decompositions

A tensor is a multidimensional array, also called an N-way array. Tensor decompositions refer to the factorization of tensors. A survey on tensor decompositions and applications is provided in [56]. In the areas of signal processing, data mining and graph analysis, tensor decompositions of higher-order tensors (i.e., $N \geq 3$) are commonly used. In this thesis, we primarily focus on two-way tensors, i.e., matrices and matrix decompositions. However, if one needs to handle relationships with more than two many-state variables, higher-order tensor decompositions can be applied for relation prediction.

TransE

TransE [14] models relationships by interpreting them as translations operating on low-dimensional embeddings of entities. The basic idea of TransE is that given a set of (s, p, o) triples, the relationship p between two entities s and o is considered as a translation of the embeddings. If a triple (s, p, o) holds, o should be the nearest neighbor of $s + p$ in the entity embedding space, while they should be far away otherwise. The model reduces the number of parameters and scales up to large knowledge bases. TransE focuses on two-way

interactions, and it can fail when modeling data where three-way dependencies between s, p and o are crucial.

RESCAL

Unlike TransE, RESCAL [79] is a model based on a three-way tensor decomposition. The model performs collective learning by learning latent representations of entities and latent representations of the relationships. RESCAL is designed to encode the inherent structure of binary relational data and is thus equivalent to a Tucker-2 decomposition [101].

DistMult

Another family of approaches learns embeddings of entities from neural networks and model relations as bilinear (or linear) mapping functions. For instance, NTNs (Neural Tensor Networks) [92] and DistMult belong to this family. In [107], DistMult refers to a special case of the basic bilinear scoring function, where the relation matrix is diagonal. The model uses weighted element-wise multiplication for modeling entity relations. Note that DistMult is limited to modeling symmetric relations due to Canonical Polyadic (CP) decomposition.

ComplEx

ComplEx [100] learns the embeddings of entities and relations by using vectors with complex values. It can model both symmetric and asymmetric relations thanks to the nature of the complex inner product. The model can be considered a valuable extension of DistMult, as its complex embeddings allow it to avoid DistMult's limitation to symmetric relations.

1.3.5 Remarks

The approaches described in Section 1.3.3 aim to describe the logical dependencies between features derived from SW data. In contrast, the matrix decomposition approach described in Section 1.3.4 and the relational graphical models (RGMs) also addressed in Section 1.3.2 predict the truth values of all basis statements (RDF-triples) in the SW. Unlike the matrix decomposition techniques, RGMs are probabilistic models where statements are represented by random variables. RGMs can be thought of as upgraded versions of regular graphical models, e.g., Bayesian networks, Markov networks, dependency networks, and latent variable models. RGMs have been developed in the context of frame-based logical representations, relational data models, plate models, entity-relationship models, and

first-order logic. Here, we attempt to relate the basic ideas of the different approaches to the SW framework.

The work on inductive databases in [85] pursues similar goals but is focused on the less-problematic data situation in relational databases. In [54], the authors describe SPARQL-ML, a framework for adding data mining support to SPARQL. SPARQL-ML was inspired by Microsoft's Data Mining Extension (DMX). In SPARQL-ML, a particular ontology for specifying the machine learning experiment is developed. The SRL methods in [54] are ILP-type approaches based on a closed-world assumption (Relational Bayes Classifier (RBC) and Relational Probabilistic Trees (RPT)). Those methods contrast with the work presented in this thesis, which maintains an open-world assumption that is more appropriate in the context of the SW. In our work, both model training and statement prediction also can be performed off-line if desired. In the case of off-line, inferred triples with their associated certainty values can be stored, e.g., in a triple store, enabling fast query execution.

Unsupervised approaches (examples of which are suitable for the relational SW domain can be found in [33, 27, 106, 52]) are flexible and interpretable, and they provide a probability distribution over a relational domain. Although unsupervised approaches are quite attractive, we fear that the sheer size of the SW and the vast number of potentially true statements it contains make such approaches inappropriate for Web-scale applications. Supervised learning, where a model is trained to make a prediction concerning a single random variable, typically shows better predictive performance and better scalability. Typical examples of supervised learning include many ILP approaches [83, 75] and propositionalized ILP approaches [23, 64]. Multivariate prediction generalizes supervised learning to predict several variables jointly, conditioned on given inputs. The improved predictive performance of multivariate prediction in comparison to simple supervised learning has been attributed to the sharing of statistical strength between multiple tasks, i.e., data is used more efficiently (see [99] and citations therein for a review). Due to the large degree of sparsity of relationship data in the SW, we expect that multivariate prediction holds great promise for SW learning, and we will apply it in this thesis. There have been a number of publications on learning with SW data, e.g., [11, 35, 93, 21, 68]. Our focus is on machine learning approaches that permit the derivation of probabilistic statements.

RGMs have been developed in the context of frame-based logical representations, relational data models, plate models, entity-relationship models, and first-order logic, but RGMs' main ideas (including Bayesian networks of node-classes and introduction of latent variables) can easily be adapted to the SW data model. One can distinguish two cases of adoption. In the first case, an RGM learns a joint probabilistic model over the full

SW or a segment of the SW. This case might be the most elegant approach since there is only one (SW-) world, and the dependencies between variables are truthfully modeled. The drawback of the first case is that the computational cost scales with the number of statements whose truth value is known or perhaps even with the number of all potentially true statements. In the second case, the sampling approach described in this thesis might prove more appropriate for large-scale applications. As an example, consider that the statistical unit is a student. Under these circumstances, a data point would not correspond to a set of features but rather to a local subgraph anchored at the statistical unit, i.e., the student. As before sampling would make the training time essentially independent of SW size. Ontological background knowledge can be integrated into machine learning, as discussed in this section. First, one can employ complete or partial materialization, which would derive statements from reasoning before training. Second, an ontological subgraph can be included in the subgraph of a statistical unit [86]. Also, note that the MLN might be particularly suitable to exploit ontological background information: ontologies can formulate some of the first-order formulas that are the basis for the features in the MLN. PRMs have been extended to learn class hierarchies (PRM-CH), which can be a basis for ontology learning.

RGM approaches typically make an open-world assumption.[10] Corresponding random variables are assumed to be missing at random, such that RGM approaches include an inherent mechanism to deal with missing data. If the assumption of data missing at random is not justified, more complex missing-data models need to be applied. As before, based on the estimated probabilities, weighted RDF-triples can be generated and added to the SW.

1.4 Structure of the Thesis

This chapter describes the standard data model in the Semantic Web and explains the reasoning tasks performed on its relational knowledge bases. Additionally, it reviews related work in the field of statistical learning for relation prediction.

Chapter 2 introduces a generic statistical framework, the *Statistic Unit Node Set (SUNS)*, where any machine learning approach can be applied, and proposes a novel multivariate model for relation prediction–*Reduced-Rank Penalized Regression (RRPP)*–that is robust, insensitive to model-specific parameters and capable of efficiently dealing with missing information. We tested the model on a social network data set to predict friendship between

[10] There are some exceptions, e.g., MLNs make a closed-world assumption during training.

persons, both in transductive and inductive settings.

Chapter 3 proposes a general kernel approach to SUNS. It utilizes the Nyström approximation technique to reduce the complexity of kernel computations. The chapter also discusses the scalability of this approach. The RRPP model presented in Chapter 2 can be considered as a special case of the kernel approach. We evaluate this approach using a data set from DBpedia.

Chapter 4 introduces a probabilistic model (the *R-model*) that relies on a relation-oriented sampling assumption and can handle n-ary relationships in a probabilistic manner. We apply the model to a social network with the aim of movie recommendations and explain how the model integrates contextual information to improve the quality of recommendations.

Chapter 5 describes the application of the SUNS approach to three real-world use cases: social media analysis, disease gene prioritization in life sciences, and a location-based personalized recommendation engine.

1.5 Contributions of the Thesis

All significant contributions of this thesis have been published in the following proceedings of conferences and journals:

1. [43] Yi Huang, Volker Tresp, Markus Bundschus, and Achim Rettinger. Scalable relational learning for sparse and incomplete domains. In *International Workshop on Statistical Relational Learning (SRL 2009)*, 2009

   I established the fundamental idea towards the SUNS framework.

2. [46] Yi Huang, Volker Tresp, and Hans peter Kriegel. Multivariate prediction for learning in relational graphs. In *NIPS 2009 Workshop: Analyzing Networks and Learning With Graphs*, 2009

   I formally defined the SUNS framework, introduced the RRPP model, and conducted the experiments. My contributions in [46] are described in Chapter 2.

3. [41] Yi Huang, Markus Bundschus, Volker Tresp, Achim Rettinger, and Hans-Peter Kriegel. Multivariate prediction for learning on the Semantic Web. In *Proceedings of the 20th International Conference on Inductive Logic Programming (ILP)*, 2010

   I extended the SUNS framework and investigated different sampling strategies in the experiments. Chapter 2 covers my contributions in [41].

4. [42] Yi Huang, Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A scalable kernel approach to learning in semantic graphs with applications to linked data. In *1st Workshop on Mining the Future Internet*, 2010

   I introduced the kernel SUNS which is described in Chapter 3.

5. [45] Yi Huang, Volker Tresp, Maximilian Nickel, Achim Rettinger, and Hans-Peter Kriegel. A scalable approach for statistical learning in semantic graphs. *Semantic Web Journal*, 5(1):5–22, 2014

   I refined the kernel SUNS and applied it to the disease gene prioritization problem which is the application described in Section 5.2.

6. [98] Volker Tresp, Yi Huang, Xueyan Jiang, and Achim Rettinger. Graphical models for relations - modeling relational context. In *KDIR 2011 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, Paris, France, 26-29 October, 2011*, pages 114–120, 2011

   I was co-creator of the R-model presented in [98] and I accomplished all empirical studies which are described in Chapter 4.

7. [7] Davide Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Yi Huang, Volker Tresp, Achim Rettinger, and Hendrik Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intelligent Systems*, 99, 2010

   I introduced inductive stream reasoning and conducted the evaluation on an application of semantic social media analysis that is described in Section 5.1.

8. [5] Marco Balduini, Irene Celino, Daniele Dell'Aglio, Emanuele Della Valle, Yi Huang, Tony Kyung-il Lee, Seon-Ho Kim, and Volker Tresp. BOTTARI: an augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. *Journal of Web Semantics*, 16:33–41, 2012

   I detailed the inductive stream reasoner in a location-based personalized recommendation engine which is the application described in Section 5.3, contributed to data collection, and achieved all experiments except for the test on the PUSH segment.

9. [6] Marco Balduini, Irene Celino, Daniele Dell'Aglio, Emanuele Della Valle, Yi Huang, Tony Kyung-il Lee, Seon-Ho Kim, and Volker Tresp. Reality mining on micropost

streams - deductive and inductive reasoning for personalized and location-based recommendations. *Semantic Web Journal*, 5(5):341–356, 2014

I designed the architecture of inductive stream reasoning, contributed to data collection, and carried out the experiments for the personalized recommendation engine. Section 5.3 describes my contributions in [6].

I am the first author and primary writer of publications 1-5. Publications 7-9 are joint works, mainly between Siemens and CEFRIEL[11].

[11]CEFRIEL is a digital innovation center located in Milano, Italy. https://www.cefriel.com/en/home

Chapter 2

Learning with the Statistical Unit Node Set (SUNS)

A knowledge base mirrors a particular domain in the real world. Facebook's social network data, for instance, reflects friendships and other relations among real persons. However, on the one hand, many ties that exist in the real world might be unknown in the knowledge base, and on the other hand, numerous relations do not exist by now but potentially would become real in the future. For example, an average Facebook user may obtain a list of users' accounts recommended that are not connected to her account yet, some of which are of her friends, whereas the others are foreign to her in real life, but she likely sends a friend request to them. In this recommendation process, many features of her account are taken into account, including her properties, e.g., age, gender, and the city she lives, her interests registered, and her behavior on Facebook.

Similar scenarios occur in many other research and application domains. Detecting and predicting missing ties of interest constitutes a significant challenge for relation prediction, which can be considered an attempt to complete an incomplete knowledge base (or supplement a relational graph, if one adopts the graphical model point of view). In this chapter, we introduce our learning approach, which relies on the statistical unit node set and thus is named the Statistical Unit Node Set (SUNS). It is a generally defined framework where any statistical learning algorithm, including matrix factorization-based algorithms, can be applied.

In Section 1.3, we reviewed global and conditional learning approaches that are suitable for relation prediction. Global models can efficiently deal with missing information, but might not scale well, while conditional models scale better, but encounter difficulties when faced with missing data. We propose SUNS as an approach that attempts to combine

global models' ability to handle missing data with the scalability of conditional models, thus maintaining the advantages of both methods, while simultaneously dealing effectively with highly sparse data.

The main contributions of this chapter are published in:

1. [43] Yi Huang, Volker Tresp, Markus Bundschus, and Achim Rettinger. Scalable relational learning for sparse and incomplete domains. In *International Workshop on Statistical Relational Learning (SRL 2009)*, 2009

2. [46] Yi Huang, Volker Tresp, and Hans peter Kriegel. Multivariate prediction for learning in relational graphs. In *NIPS 2009 Workshop: Analyzing Networks and Learning With Graphs*, 2009

3. [41] Yi Huang, Markus Bundschus, Volker Tresp, Achim Rettinger, and Hans-Peter Kriegel. Multivariate prediction for learning on the Semantic Web. In *Proceedings of the 20th International Conference on Inductive Logic Programming (ILP)*, 2010

This chapter is based on these publications but is mostly rewritten. Section 2.1.5 explains the RRPP model in more details than [46]. Moreover, Section 2.2 extends the empirical study by adding four further settings 5-8 that are novel and have not been published yet. The experiments in [43] are not included in this thesis.

This chapter is structured as follows. First, we introduce the approach SUNS in two parts: One part is the definition of a general statistical setting, and the other is a novel multivariate prediction model. Then, we investigate SUNS on a data set gathered from a social network. In the end, we conclude the chapter with some discussions and remarks about the approach.

2.1 The Approach

2.1.1 Definition of Statistical Setting

In a relational domain, the issue of a probability distribution is somewhat troublesome. Considering a set of physicians and patients, let Q be the quality of a physician and assume that half of the physicians are highly competent, while half are not. If we randomly sample physicians sufficient times, the number of the skilled physicians and the poorly competent ones should be nearly equal $P(Q = Good) \approx P(Q = Bad) \approx 0.5$. However, patients might prefer to visit good physicians. Thus, if we sample patients, we might obtain another probability distribution of physicians, for instance, $P(Q = Good) = 0.7$,

meaning that 70% of the physicians whom the sampled patients have visited are good. Thereby we can conclude that in relational domains, the interpretation of a probability distribution is determined by the sampling process. More precisely, the definition of a proper statistical setting is crucial, including the definition of the statistical units, the population, the sampling procedure as well as associated features.

- *A statistical unit* is an entity or instance of a certain type, e.g., person. Statistical units are the source of the variables or features of interest.
- *A population* is a set of statistical units under consideration. A population might be defined in various ways. For example, it might include all persons living in a particular country or all-female students studying at a particular university.
- *A sample* is a subset of the population. In a statistical analysis, only a sample is made available for investigation. In the experimental section, we will explore various sampling strategies. Random sampling is a typical approach, as in the previous example of physicians. Based on the sample, a data matrix is generated where the statistical units define the rows.
- *Features* are the common properties of the sampled statistical units. They define the columns of the data matrix.

In the following two sections, we informally explain how the features–random variables and non-random covariates–are generated.

2.1.2 Random Variables

In Section 1.2 we described the Semantic Web where relational knowledge bases are published, shared, exchanged, and widely exploited. We also learned that knowledge bases in the Semantic Web are represented as RDF graphs, or a collection of RDF triples in the form of (s,p,o) where subject s and object o are related via predicate p. We now introduce for each actual or potential triple a *triple node* drawn as a diamond-shaped node in Figure 1.1. A triple node is in state *one* (*true*) if the triple is known to exist and is in state *zero* (*false*) if the triple is known not to exist. Graphically, one only draws the triple nodes in state *one*, i.e., the existing triples.

In the following, we now associate triples with statistical units, with the aim of assigning a triple to a statistical unit if the statistical unit appears in the triple. Let us consider the statistical unit *Jane*. Based on the triples she is participating in, we obtain

three expressions *(?personA, rdf:type, Person)*, *(Joe, knows, ?personA)*, and *(?personA, hasIncome, High)* where *?personA* is a variable representing any statistical unit. These expressions form the random variables and define columns in the data matrix.[1] By considering the remaining statistical units *Jack* and *Joe* we additionally generate the expressions *(?personA, knows, Joe)*, *(Jack, knows, ?personA)*, and *(?personA, knows, Jane)*. We will not add *(Jane, knows, ?personA)* since Jane considers no one in the knowledge base to be her friend. We iterate this procedure for all statistical units in the sample and add new expressions (i.e., columns in the data matrix) if necessary. Note that expressions that are not present in the sample will not be considered. Also, expressions that are rarely true (i.e., associated with very few statistical units) are removed, since no meaningful statistics can be derived from uncommon occurrences.

In [97], the triples associated with a statistical unit were denoted as a *statistical unit node set* (SUNS). The matrix formed with n statistical units as rows and the random variables as columns is denoted as Y.

2.1.3 Non-Random Covariates

The features we have derived so far represent truth values of actual or potential triples. Those triples are treated as random variables in our analysis. If a machine learning algorithm predicts that a triple is very likely, we can enter this triple in the knowledge base from which the triples are retrieved, and add columns that provide additional information, but are treated as covariates or fixed features.

First, we derive a type of generalized relations from the sample. More precisely, we consider the expressions derived in the last section and replace constants with variables. For example, from *(?personA, knows, Jane)* we derive *(?personA, knows, ?personB)* and count how often this expression is true for a statistical unit *?personA*. In the example, we count the number of friends of person *?personA*.

Second, we consider expressions aggregated by examining triples outside of a statistical unit node set. The example contains a binary triple *(?personA, knows, Jane)*. If Jane is part of another binary triple, saying *(?personA, hasIncome, High)*, then we replace Jane with a variable *personB*, form a composite expression (*?personA, knows, ?personB)* $\wedge$ *(?personB, hasIncome, High)*, and count how many rich friends a person has. It is possible to derive many kinds of additional features in different ways, but so far, we have restricted ourselves to these two types. The matrix formed with n statistical units as rows and fixed

[1] Don't confuse a random variable representing the truth value of a statement with a variable in a triple, representing an object.

features as columns is denoted as X.

After constructing the data matrix, we prune away columns in X and in Y which have *ones* in fewer than ϵ percent of all rows, where ϵ is a small number, because, as discussed previously, no meaningful statistical analysis is possible for those features. Note that by applying this pruning, we usually can reduce an exponential number of features to a reasonably small number.

2.1.4 Formal Definition of Statistical Setting

Let $U = \{u_i\}_{i=1}^{n}$ be the set of statistical units in the sample under consideration. First, we define a statistical node set SUNS_{u_i} for statistical unit u_i to include all triple nodes that correspond to all actual and potential triples in which u participates either as the subject or as the object. In the process, we must apply a restriction: if there is a triple between the statistical units $u_i, u_j \in U$ of the form (u_i, p, u_j), then the triple is a member of SUNS_{u_i} but not a member of SUNS_{u_j}. Otherwise, the same triple node would appear in two different node sets, which would make the two sets highly dependent. In Figure 2.1, $\{a_i\}$ belongs to statistic unit A, but not to B.

The resulting data matrix contains one row for each statistical unit, and we now define its features as follows.

1. Let (p, o) be a pair, such that a triple of form (u, p, o) is in the knowledge base (KB), for at least one $u \in U$. For each distinct (p, o), a column is generated in the data matrix. The entry in the data matrix for statistical unit u and pair (p, o) is equal to *one* if the triple (u, p, o) exists in the KB, and is *zero* otherwise.

2. In addition, we generate a column for each distinct p. The entry in the data matrix for statistical unit u and property (predicate) p is equal to *one* if the triple (u, p, o) exists for at least one o in the KB, and is *zero* otherwise.

3. Let (s, p) be a pair, such that a triple of the form (s, p, u) is in the KB for at least one $u \in U$. For each distinct (s, p), we generate a column in the data matrix. The entry in the data matrix for statistical unit u and pair (s, p) is equal to *one* if the triple (s, p, u) exists in the KB, and is *zero* otherwise.

4. In addition, we generate a column for each distinct p. The entry in the data matrix for statistical unit u and property p is equal to *one* if the triple (s, p, u) exists for at least one s in the KB, and is *zero* otherwise.

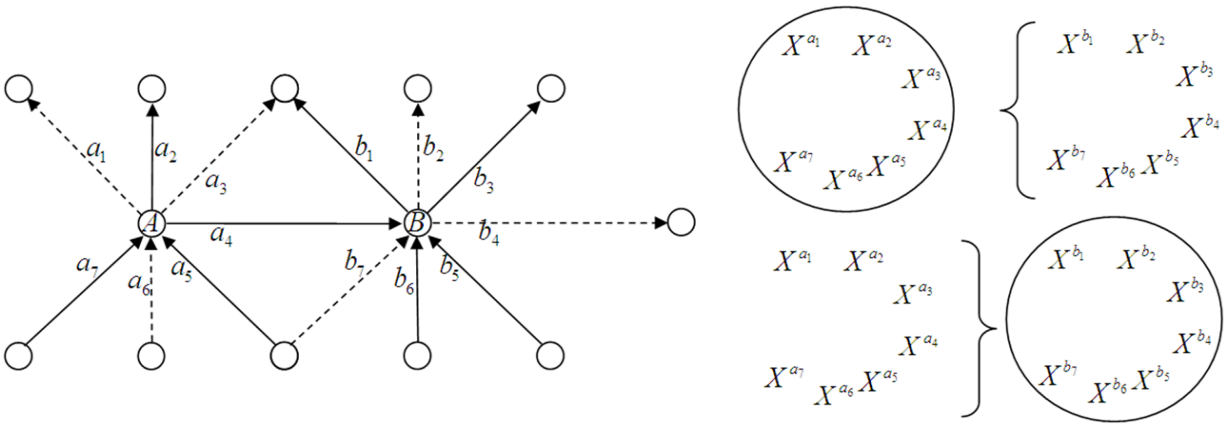


Figure 2.1: Left: An RDF-graph fragment with two statistical units A and B. $\{a_i\}$ are triples assigned to A and $\{b_i\}$ are triples assigned to B. Dashed lines indicate triples that are not in the knowledge base. Right top: The probabilistic nodes in the circle form the SUNS for a statistical unit A and are modeled jointly. Information from triples not in a SUNS (here, the states of the probabilistic nodes $\{X^{b_i}\}$) are considered to be input features of A. Right bottom: The probabilistic nodes in the circle form the SUNS for a statistical unit B and are modeled jointly. Information from triples not in a SUNS (here, the states of the probabilistic nodes $\{X^{a_i}\}$) are considered to be input features of B.

Due to the restriction mentioned above, if there are triples between the statistical units of the form (u_i, p, u_j) with $u_i, u_j \in U$, we remove the columns for u_j where a statistical unit u_j acts as an object. This guarantees that a particular expression appears only once in the data matrix.[2] Note that because of this restriction, the resulting number of columns is smaller than we described in Sections 2.1.2 and 2.1.3. The example in Figure 1.1 illustrates that the expressions *(Joe, knows, ?personA)* and *(Jack, knows, ?personA)* have to be removed. Since fixed covariates are generated similarly, we do not repeatedly describe its generation process.

Once the data matrix is generated, one can predict potential relations not only in the sample through transduction but also for all other statistical units in the population through an inductive process. In the experiment section, we will describe these two settings (i.e., transduction and induction) more precisely.

[2]This is not the only possible way to generate a data matrix, but an important feature is that only triple nodes within a SUNS, are evaluated.

2.1.5 Multivariate Prediction Model: Reduced-Rank Penalized Regression

In previous sections, we proposed a statistical setting where the statistical unit node set (SUNS) is defined mainly based on local neighborhoods of statistical units. By adding aggregated information derived from the neighborhood, homophily can be modeled as well. For instance, the income of a person can be predicted by the average income of this person's friends.

As we will see in our experiments, data matrices representing samples from knowledge bases are typically high-dimensional and sparse. Multivariate prediction approaches have been most successful [99] at exploiting this type of matrix. In a multivariate prediction model, all outputs (i.e., the random variables) are predicted jointly such that statistical strength can be shared between outputs. This is the case because some or all model parameters are sensitive to all outputs, which improve the estimates of those parameters.[3] Theoretically, any learning algorithm can be applied in this statistical setting, but we focus on multivariate prediction algorithms in particular.

Traditionally, regression problems involve inputs $X = [x_1; ...; x_n] \subset \mathbb{R}^{n \times m}$ and corresponding outputs $y = [y_1; ...; y_n]$, the states of the target variable var. X is a n-by-m data matrix, and y is a column vector with n rows. y_i is estimated based on each row vector $x_i = (x_{i,1}, ..., x_{i,m}) \in X$. In our case, $y_i \in [0, 1]$ represents the probability of the state being true, i.e., $y_i = P(var = true|x_i)$, and we start with a classical linear regression assuming a linear dependence between y_i and x_i

$$y_i = f(x_i) = x_i w^\top + \epsilon_i \tag{2.1}$$

where w is a vector of weights and $\epsilon_i \sim N(0, \sigma^2)$ is the noise or error which is normal distributed with mean 0 and variance σ^2. For all inputs X and all outputs y, we rewrite Equation (2.1) as

$$y = f(x) = Xw^\top + \epsilon \tag{2.2}$$

The goal is to learn such a $\hat{w}$ that the estimated outputs $\hat{y} = Xw^\top$ approximate the actual outputs y as much as possible, thereby minimizing error ϵ.

$$\hat{w} = \underset{w}{argmin}\{\ell(y, \hat{y})\} \tag{2.3}$$

[3] Although the completion is applied to the entire matrix, only *zeros* —representing triples with unknown truth values— are overwritten.

When we assume that the input data is independent and identically distributed, and when we apply the least square cost function, we obtain

$$\ell(y, \hat{y}) = \sum_{i=1}^{n} (y_i - x_i w^\top)^2 = (y - Xw^\top)^\top (y - Xw^\top) \tag{2.4}$$

We find the gradient vector by calculating the derivative of ℓ with respect to w. That results in the equation.

$$\hat{w} = (X^\top X)^{-1} X^\top y \tag{2.5}$$

Applying a regularized linear regression model adds a regularization term $\|w\|$ in order to avoid overfitting problem, and results in

$$\hat{w} = \underset{w}{argmin}\{-\ell(y, \hat{y}) + \lambda \|w\|\} \tag{2.6}$$

where $\|\cdot\|$ is the L2-norm and λ makes a tradeoff between the error and the complexity of the model. Equation (2.5) is extended as

$$\hat{w} = (X^\top X + \lambda I)^{-1} X^\top y \tag{2.7}$$

After training we multiply $\hat{w}$ by X to approximate y (analogously one multiplies $\hat{w}$ with a new data set X_{new} to predict states of the target variable $\hat{y}_{new}$).

$$y \approx \hat{y} = X(X^\top X + \lambda I)^{-1} X^\top y \tag{2.8}$$

In the statistical setting defined above there are multiple outputs Y, i.e., the random variables. Y is a n-by-l matrix containing n rows and l columns. Based on an input vector x_i the outputs $y_{i,j}$ for $j = 1, .., l$ are predicted. We estimate the states of all variables and, analogously to Equation (2.5), we learn a l-by-m weighting matrix $\hat{W} = [\hat{w}_1; ..; \hat{w}_l]$.

$$\hat{W} = (X^\top X + \lambda I)^{-1} X^\top Y \tag{2.9}$$

Then we approximate Y in the same way we have done in Equation (2.8)

$$Y \approx \hat{Y} = X(X^\top X + \lambda I)^{-1} X^\top Y \tag{2.10}$$

Unlike Equation (2.8), Equation (2.10) learns the parameter matrix $\hat{W}$ by trying to fit all outputs *jointly* rather than independently.

The aim of our multivariate algorithm is to consider inputs together with outputs when learning $\hat{W}$. In other words, $\hat{W}$ must be learned to fit not only Y, but also X. As a

byproduct, we can approximate X as well. We define $D = [XY]$ concatenating X and Y column-wise, then we replace both X and Y in Equations 2.9 and 2.10 by D.

$$\hat{W} = (D^\top D + \lambda I)^{-1} D^\top D \tag{2.11}$$

$$D \approx \hat{D} = D(D^\top D + \lambda I)^{-1} D^\top D \tag{2.12}$$

We introduce a novel multivariate prediction model that we call Reduced-Rank Penalized Regression (RRPP). First, we apply Singular Value Decomposition to D.

$$D = USV^\top \tag{2.13}$$

U and V are left and right singular vectors, respectively. U is a $n \times n$ matrix and V is a $m \times m$ matrix. Note that both U and V are orthogonal and quadratic. S is an $n \times m$ matrix containing positive numbers on the diagonal, i.e., the singular values of D, and *zeros* somewhere else. From this, we can derive the following equations.

$$D^\top D = VS^2V^\top \tag{2.14}$$

$$U = DVS^{-1} \tag{2.15}$$

$$V^\top = U^\top DS^{-1} \tag{2.16}$$

Then, by replacing replace D and $D^\top D$ in Equation (2.12), we arrive at

$$\begin{aligned} \hat{D} &= USV^\top (VS^2V^\top + \lambda I)^{-1} VS^2V^\top \\ &= USV^\top (V(S^2 + \lambda I)V^\top)^{-1} VS^2V^\top \\ &= USV^\top (V^\top)^{-1}(S^2 + \lambda I)^{-1}(V)^{-1} VS^2V^\top \\ &= U\text{diag}(\frac{s^3}{s^2 + \lambda})V^\top \end{aligned} \tag{2.17}$$

where diag$(\cdot)$ stands for the diagonal elements (i.e., the singular values) in S.

Finally, we obtain the approximation $\hat{D}$ by taking only the r largest singular values and the r corresponding singular vectors denoted as U_r and V_r, where $r \ll rank(D) \leq min(n, m)$.

$$\hat{D} = U_r \text{diag}_r(\frac{s^3}{s^2 + \lambda})V_r^\top \tag{2.18}$$

Note that the rank of $\hat{D}$ is reduced to $\leq r$, and the number of the singular values in diag_r is also reduced to r.

2.1.6 Transduction and Induction

In this process, both U_r and V_r might not necessarily be held in memory. Instead, V_r alone might be used to make predictions in the primal form derived from Equation (2.15).

$$\hat{D} = DV_r\text{diag}_r(\frac{s^2}{s^2+\lambda})V_r^\top \tag{2.19}$$

Alternatively, we might use only U_r in the dual form derived from Equation (2.16).

$$\hat{D} = U_r\text{diag}_r(\frac{s^2}{s^2+\lambda})U_r^\top D \tag{2.20}$$

It is important to note that the original data D is usually a large but very sparse matrix. In contrast, $\hat{D}$ is a dense matrix where previously missing entries are fulfilled by the estimated likelihoods of these entries being true. For the same reason, it is not necessary in practice–and indeed it is sometimes even impossible–to hold $\hat{D}$ in memory. Instead, we could approximate the data vectors in $\hat{D}$ one by one, [4] rather than all of the data in one step, as in Equation (2.18). Using the primal form, approximations for one data vector $d_i = [x_i y_i]$ can be made as follows:

$$\hat{d_i} = d_i V_r\text{diag}_r(\frac{s^2}{s^2+\lambda})V_r^\top \tag{2.21}$$

So far, we have discussed how to learn $\hat{W}$ from data D and how to make the approximations $\hat{D}$. That process can be understood as a means of filling in missing facts in a sample by transferring knowledge–in this case, $\hat{W}$–learned from the known facts the very same sample. That process, therefore, constitutes a form of transduction (also mentioned in Section 2.1.4). Alternatively, when applying $\hat{W}$ onto a new data vector d_{new} outside of the sample (but in the population), just as traditional predictive models do, the process is one of induction, and the primal form of this inductive process is formulated as follows:

$$\hat{d}_{new} = d_{new} V_r\text{diag}_r(\frac{s^2}{s^2+\lambda})V_r^\top \tag{2.22}$$

[4] The block-wise approximation is certainly also applicable.

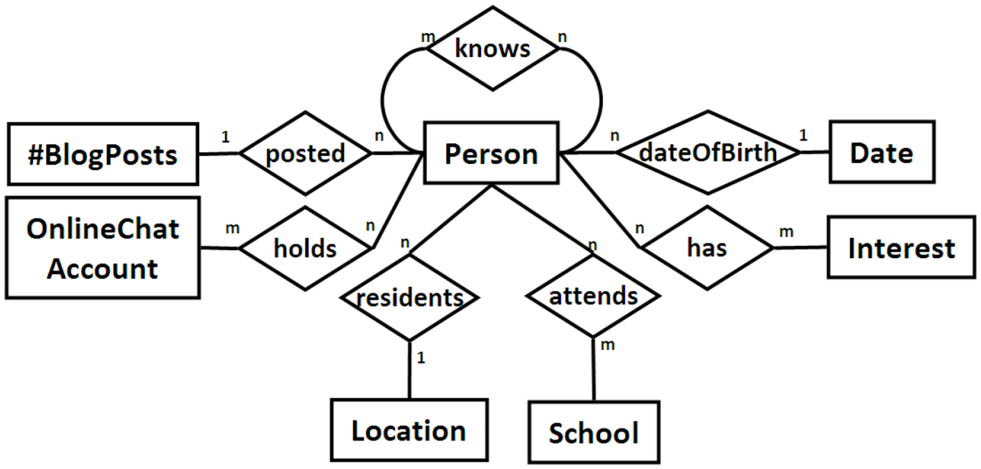


Figure 2.2: Entity-relationship diagram of the LJ-FOAF domain

where d_{new} is a row vector. Analogously, the dual form is derived by using Equation (2.16)

$$\begin{aligned}\hat{d}_{new} &= d_{new}(U_r^\top DS^{-1})^\top \text{diag}_r(\frac{s^2}{s^2+\lambda})U_r^\top DS^{-1} \\ &= d_{new}D^\top U_r \text{diag}_r(\frac{1}{s^2+\lambda})U_r^\top D\end{aligned} \tag{2.23}$$

Equation (2.11) shows that $\hat{W}$ is a $m \times m$ dense matrix and requires memory of size m^2. For any $m > 100.000$, $\hat{W}$ hardly fits into memory. The primal form and the dual form reduce the memory complexity and allow us to make predictions efficiently, and also to efficiently persist $\hat{W}$ if necessary, because U_r is a $n \times r$ matrix, V_r is a $r \times m$ matrix and r is usually a reasonably small constant, for instance, $r = 50$. Using U_r or V_r, the demand for memory is linearly dependent on n or m respectively. The choice between the primal form and the dual form should be made according to the dimensions of the data matrix D, more precisely, in case $n \gg m$, the dual form performs more efficiently than the primal, while $m \gg n$, the primal form is preferred.

2.2 Empirical Study: Friendship Prediction

2.2.1 Data Set and Experimental Setup

Evaluation Procedure and Evaluation Measure

In this study, the task is to predict a person's potential friends, i.e., *knows* statements. For each person in the data set, we randomly selected one *knows* friendship statement to use as

a test statement, setting the corresponding matrix entry to *zero* in order for it to be treated as unknown. In the test phase, we then predict the probability of all unknown friendship entries, including the (zeroed) entry of the test statement, which should, in principle, be assigned a higher likelihood of being true than other unknown friendship entries in the data set.

We use the normalized discounted cumulative gain (NDCG) method (described in Appendix 6) to evaluate a ranking of predictions generated by approaches we evaluate. These NDCG scores are averaged over all random variables for comparison. The better an algorithm, the higher the NDCG scores, and the higher the test statement is placed in the ranking of predictions.

Data Set

Our friendship prediction experiments were carried out using a friend-of-a-friend (FOAF) data. The purpose of the FOAF project [16] is to create a web of machine-readable pages describing people, their relationships, and people's activities and interests, using W3C's RDF technology. The FOAF ontology is based on RDFS/OWL and is formally specified in the FOAF Vocabulary Specification 0.91[5].

We gathered our FOAF data set from user profiles of the community website LiveJournal.com[6]. The extracted entities and relations between them are shown in Figure 2.2. In total, we collected data comprising 32,062 persons and their related attributes. An initial pruning step removed persons with very few connections to others, as well as especially rare attributes. Table 2.1 lists the number of different individuals (top rows) and their known instantiated relations (bottom rows) in the full triple set, in the pruned triple set, and in triples sets in different experimental settings(explained below). After pruning, the resulting data matrix contained 14,425 rows (persons) and 15,206 columns. Among those columns, 14,425 *one* values (friendship attributes) refer to the property *knows*. The remaining 781 columns (general attributes) refer to general information about age, location, number of blog posts, the school an individual attended, his or her online chat account, and interests.

Algorithms

We create a random ranking for all unknown triples, denoted as *Baseline*, in which every unknown triple is assigned a randomly generated probability. Additionally, we assume

[5]http://xmlns.com/foaf/spec/

[6]http://www.livejournal.com/bots/

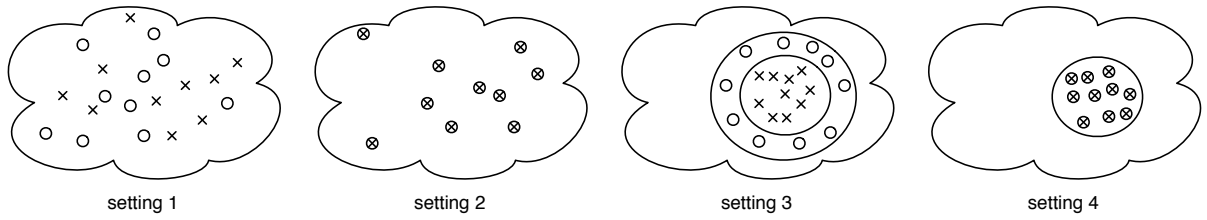


Figure 2.3: Evaluated sampling strategies

that the friends of friends of a particular person might be his or her friends as well. That assumption leads to another baseline method, denoted as *FOF, d=2*, meaning second-degree friendship. From the RDF graph point of view, we propagate the *knows* relation one step further alongside existing *knows* edges.

In addition to our RRPP approach, we investigate three state-of-the-art multivariate prediction approaches: singular value decomposition (SVD), non-negative matrix factorization (NNMF) [65], and latent Dirichlet allocation (LDA) [12]. All of these approaches estimate unknown matrix entries via a low-rank matrix approximation. NNMF performs a decomposition under the constraint that all entries in the factorized matrices are non-negative. LDA is a generative topic model and decomposes the data matrix based on a Bayesian treatment. After matrix completion, entries are interpreted as the likelihoods with which the corresponding triples might be true, and we focus on those entries which previously were *zeros*.

Data Retrieval and Sampling Strategies

In our experiments, we evaluated the generalization capabilities of the multivariate prediction learning algorithms mentioned above, using eight different settings. The first four settings are illustrated in Figure 2.3, in which a cloud symbolizes the part of the Web that can easily be accessed (in our case, the data set given in Table 2.1). Crosses represent persons who are known during the training phase (the training set), and circles represent persons whose *knows* relations need to be predicted.

Setting 1 describes a case where the depicted part of the knowledge base is randomly accessible, meaning that all instances can be queried directly from triple stores. Statistical units in the training sample are randomly sampled, and statements for other randomly selected statistical units are predicted for testing (inductive setting). In this setting, persons are rarely connected by the *knows* relations. The *knows* relation

in the training set and test set is very sparse (0.18%).

Setting 2 also illustrates a case where statistical units in the sample are randomly selected, but this time the truth values of statements concerning the statistical units in the training sample are predicted (transductive setting). Some instances of the *knows* relation of the selected statistical units are withheld from the training sample and used for prediction. Compared to setting 1, predictions should be more comfortable in this setting, since the statistics for training and prediction match perfectly.

Setting 3 assumes that the Web address of one user (i.e., one statistical unit) is known. Starting from this random user, we identified other users connected by the *knows* relation by breadth-first crawling, who are then added to the training set. The test set is constituted by continuing this crawling (inductive setting). In this way, all person profiles are fully connected, whether directly or indirectly. The training profiles show higher connectivity (1.02%) than the test profiles (0.44%). In this situation, generalization can be expected to be more effective than in settings 1 and 2 since local properties are more consistent than global ones.

Setting 4 combines settings 2 and 3. In this setting, truth values of statements concerning the statistical units in the training sample are predicted (transductive setting). Instances of the *knows* relation are withheld from training and used for prediction.

Settings 5-8 use the same sets of statistical units as settings 1-4, respectively. However, whereas the data matrices of settings 1-4 only contain friendship relations to persons *within* the sample, friendship relations in settings 5-8 are associated with any person in the *population*, which may include some individuals outside of the sample. In settings 5-8, we remove users' friends (friendship attributes) who are known by less than ten users (statistical units). As a result, these settings' data matrices are denser when compared to the matrices in settings 1-4. The concrete numbers of statistical units and friendship attributes are shown in *Person* (row) and *Person* (col) respectively in Table 2.1.

In settings 1 and 2, we randomly sampled 2,000 persons to generate a training set. In setting 1, we again randomly sampled 2,000 *other* persons for the test set. In setting 3, 4,000 persons were sampled, the first half of which was used for training and the second half for testing. Setting 4 required only those 2,000 persons in the training set in setting 3. In settings 5-8, we followed the same sampling strategies as in settings 1-4, respectively, but then extracted all users known by the sampled users in order to define a set of friendship

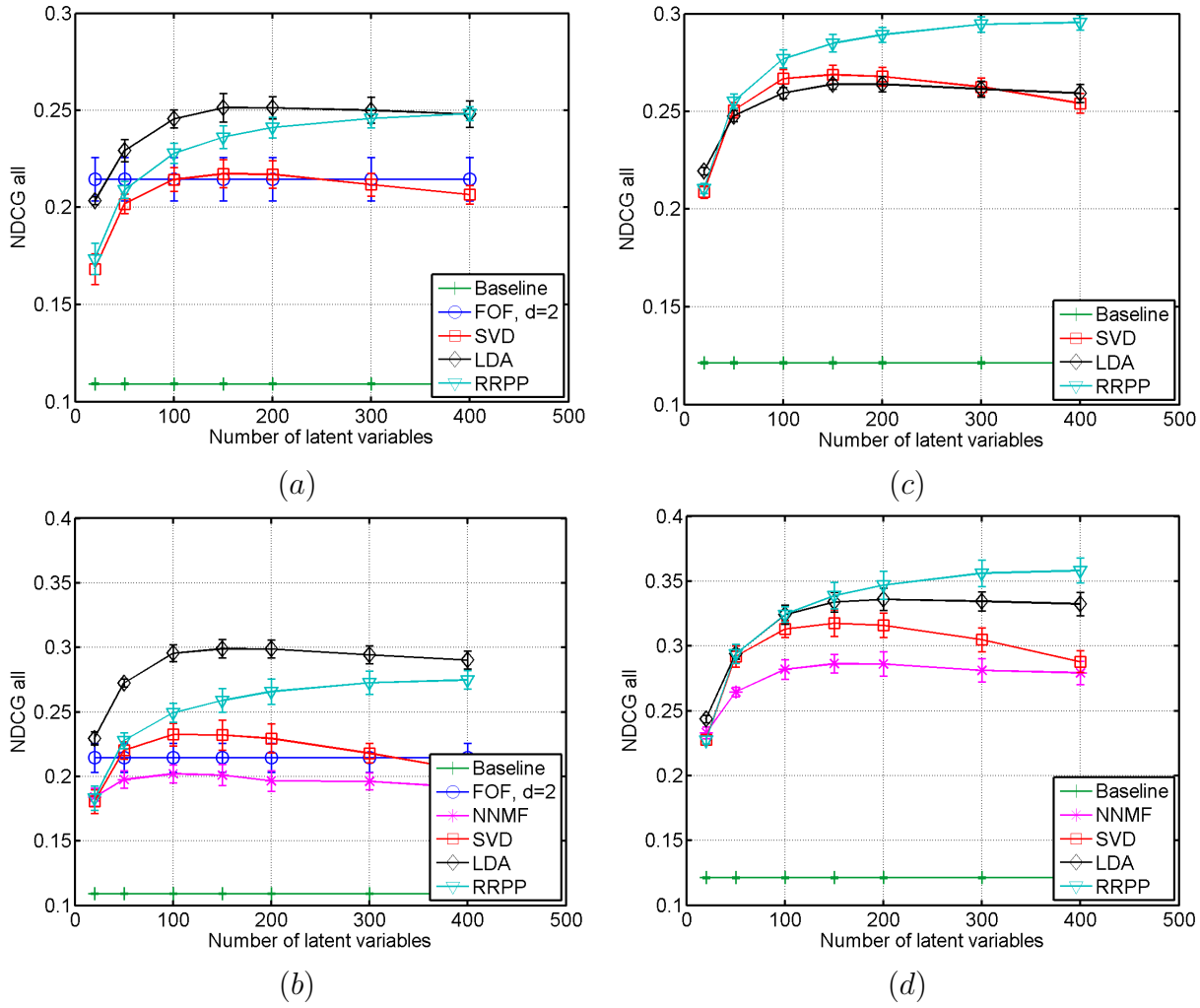


Figure 2.4: Comparison between different algorithms. *NDCG all* is plotted against the number of latent variables: (a)-(d) for settings 1, 2, 5, and 6 respectively.

attributes. In each case, the sampling was repeated five times, such that error bars could be derived. Table 2.1 reports details of the samples (training set and, if applicable, test set).

2.2.2 Results

The two baseline methods and the four multivariate prediction approaches proposed in Section 2.1.5 were applied to the training set, repeating the evaluation procedure described above ten times for each sample. Since NNMF is only applicable in a transductive setting, it was not evaluated in settings 2, 4, 6, and 8. The *FOF, d=2* was not applied in settings

		full	pruned	setting 1		setting 2	setting 3		setting 4	setting 5		setting 6	setting 7		setting 8
				training	test		training	test		training	test		training	test	
Concept	*Person* (row)	32,062	14,425	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000
#Indivi.	*Person* (col)	-	-	2,000	2,000	2,000	2,000	2,000	2,000	1,122	1,122	1,122	1,297	1,297	1,297
	Location	5,673	320	320	320	320	320	320	320	320	320	320	320	320	320
	School	15,744	329	329	329	329	329	329	329	329	329	329	329	329	329
	Interest	4,695	118	118	118	118	118	118	118	118	118	118	118	118	118
	On.ChatAcc.	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	Date	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	#BlogPosts	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Role	*knows*	530,831	386,327	7,311	7,339	7,339	40,786	17,613	40,786	14,909	16,869	16,869	41,705	18,595	41,705
#Inst.	(*sparsity*)	0.05%	0.19%	0.18%	0.18%	0.18%	1.02%	0.44%	1.02%	0.66%	0.75%	0.75%	1.61%	0.72%	1.61%
	residence	24,368	7,964	1,122	1,106	1,106	1,172	1,217	1,172	1,122	1,106	1,106	1,172	1,217	1,172
	attends	31,507	5,088	676	747	747	718	749	718	676	747	747	718	749	718
	has	9,616	1,659	206	246	246	216	208	216	206	246	246	216	208	216
	holds	19,021	8,319	1,134	1,087	1,087	1,168	1,075	1,168	1,134	1,087	1,087	1,168	1,075	1,168
	dateOfBirth	10,040	5,287	777	715	715	779	784	779	777	715	715	779	784	779
	posted	31,959	14,369	1,993	1,992	1,992	1,994	1,991	1,994	1,993	1,992	1,992	1,994	1,991	1,994

Table 2.1: Number of individuals and number of instantiated relations in the original data set and in the pruned data set (see text) as well as the statistics about the different experimental settings

Method	setting 1	setting 2	setting 3	setting 4	setting 5	setting 6	setting 7	setting 8
Baseline	0.1092 ± 0.0003	0.1092 ± 0.0003	0.1094 ± 0.0001	0.1094 ± 0.0001	0.1213 ± 0.0005	0.1213 ± 0.0005	0.1216 ± 0.0042	0.1216 ± 0.0042
$FOF, d = 2$	0.2146 ± 0.0095	0.2146 ± 0.0095	0.1495 ± 0.0077	0.1495 ± 0.0077	NaN	NaN	NaN	NaN
NNMF	NaN	0.2021 ± 0.0058 r=100	NaN	0.2983 ± 0.0197 r=150	NaN	0.2864 ± 0.0067 r=150	NaN	0.3217 ± 0.0403 r=100
SVD	0.2174 ± 0.0061 r=150	0.2325 ± 0.0074 r=100	0.2085 ± 0.0147 r=200	0.3027 ± 0.0179 r=100	0.2688 ± 0.0044 r=150	0.3176 ± 0.0092 r=150	0.2407 ± 0.0413 r=100	0.3411 ± 0.0179 r=50
LDA	0.2514 ± 0.0049 r=200	0.2988 ± 0.0057 r=200	0.2288 ± 0.0123 r=200	0.3374 ± 0.0117 r=200	0.2640 ± 0.0022 r=150	0.3359 ± 0.0079 r=200	0.2331 ± 0.0143 r=150	0.3470 ± 0.0372 r=200
RRPP	0.2483 ± 0.0018 r=400	0.2749 ± 0.0037 r=400	0.2252 ± 0.0049 r=400	0.3315 ± 0.0109 r=400	0.2956 ± 0.0019 r=400	0.3582 ± 0.0049 r=400	0.2607 ± 0.0088 r=400	0.3591 ± 0.0237 r=400

Table 2.2: Best *NDCG all* averaged cross over the samples with 95% confidence interval where r stands for the number of latent variables

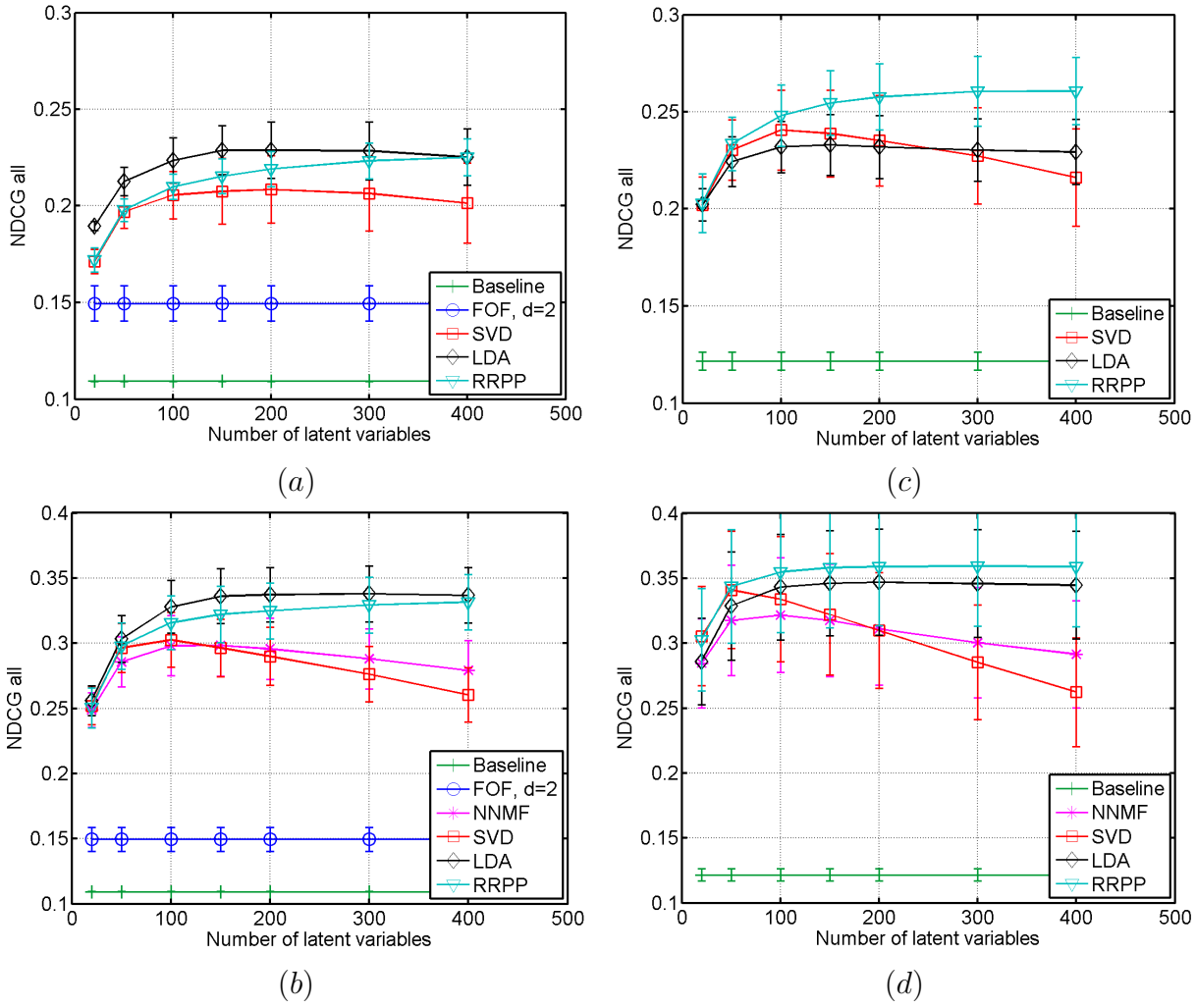


Figure 2.5: Continue Figure 2.4: (a)-(d) for settings 3, 4, 7, and 8 respectively.

5-8, because it was impossible for many users to access the friends of their friends outside of the generated sample. Consequently, for the algorithms in those settings, no results were reported in the figures or in Table 2.2 (i.e., NaN).

Figures 2.4 and 2.5 show the experimental results achieved on our FOAF data set. The error bars indicate 95% confidence intervals based on the standard error of the mean over the samples. The *NDCG all* scores of the algorithms against the number of latent variables are plotted in Figure 2.4 for inductive settings 1, 2, 5, and 6, and in Figure 2.5 for transductive settings 3, 4, 7, and 8. The best *NDCG all* scores of all algorithms in different settings are recorded in Table 2.2, where r indicates the number of latent variables achieving the best scores.

First, we observe that the experimental results in settings 5-8 are remarkably better than those in settings 1-4. This can be attributed to the fact that, in settings 5-8, loosely connected friends were pruned, and consequently, a more dense friendship network was established.

Additionally, all multivariate prediction algorithms outperform the baseline algorithms in all settings except for NNMF and SVD, which are only slightly better than FOF, d=2 in settings 1 and 2.

Furthermore, we observe that LDA and RRPP outperform NNMF and SVD in each setting, and that they are not sensitive to the number of latent variables so long as that number is reasonably large. For instance, LDA reaches its best NDCG score with $r = 150$ latent variables in setting 4, and its performance does not deteriorate when the number of latent factors increases. RRPP's score also increases with the number of latent variables, over the range of such numbers our experiments contained. In contrast, NNMF and SVD are sensitive to the number of latent variables.

Comparing the results across different settings, we can also observe that the multivariate prediction approaches perform best in setting 4, followed by setting 2, then setting 1, and perform worst in setting 3. A similar result can be seen in settings 5-8 as well. By way of comparison, the random guess baseline achieves (almost) the same score in all settings. The multivariate approaches' higher scores in settings 4 and 8 suggest that, in general, a link-following sampling strategy performs better than random sampling. Similar results in statistical comparisons between random and link-following sampling have been obtained in other works, e.g., [77].

Finally, we note that prediction performance in setting 1 is only slightly worse than that in setting 2, while prediction performance in setting 4 is significantly better than in setting 3. The same phenomenon occurs in settings 5-8. We attribute this to the fact that, in settings 3 and 7, the density of friendship relations in the training set and test set are very different. In Table 2.1, it is apparent, for instance, that the *knows* relations in the training data set (1.02%) used in setting 3 are significantly more dense than in the test data set (0.44%). Intuitively, it would seem that the people in the training set know each other quite well, whereas the people in the test set know the people in the training set less well.

2.3 Remarks

In this chapter, we first introduced a well-defined statistical setting–SUNS–where data matrices can be generated in a methodical way, including random variables and fixed covariates based on the neighborhoods of statistical units. We then proposed a novel multivariate algorithm for relation prediction, RRPP, which has two primary advantages over other approaches: first, it can efficiently learn a global model that exploits all features of statistical units and handles missing values naturally; second, the algorithm combines regularization and low-rank matrix factorization, which makes it generalizable to statistical units inside and outside of a sample. In our experiments, we tested the new algorithm on a social network data set to predict friendship between persons in both transductive and inductive settings. Compared to other benchmark methods, RRPP demonstrated promising predictive performance and proved to be highly generalizable. In contrast, other approaches based on single predictions, such as SVMs, failed to perform as well as our baseline approaches, and were not reported as a result.

Our algorithm has two parameters: 1) the reduced rank r of data matrices and 2) the tradeoff λ between regularization and error. In general, both parameters can be tuned through cross-validation. However, heuristically, one might possibly set r to a reasonably large number, since PPRR is not sensitive to r thanks to its strong generalizability, and choose a λ value proportional to the largest singular value of the data D. In our experiments, LDA also outperformed SVD and NNMF. LDA's superior performance can be explained by the fact that LDA is based on a Bayesian treatment. Consequently, we would recommend RRPP or LDA as default methods due to their insensitivity to precise parameter tuning.

Our proposed approach can be extended in many ways, including by allowing the user to specify additional parameters in the learning process, if desired, along the lines of the extensions described in [54]. Our approach might also be extended in connection with ontological background knowledge. So far, ontological background knowledge was taken into consideration only by including logically inferred statements into learning.

Finally, learned probabilistic statements generated by our algorithm can be queried in order to provide recommendations. The following SPARQL query illustrates a search for LiveJournal users who live in Munich and might want to be Trelena's friend:

```
1  PREFIX ya:   http://blogs.yandex.ru/schema/foaf/
2  PREFIX foaf: http://xmlns.com/foaf/0.1/
3  PREFIX dc:   http://purl.org/dc/elements/1.1/
4  SELECT DISTINCT ?person
5  WHERE { ?person ya:located ?city .
6          ?person foaf:knows <http://trelana.livejournal.com/trelana>
7              WITH PROB ?prob .
8          FILTER REGEX(?city, "Munich") .
9  }
10 ORDER BY DESC(?prob)
```

Listing 2.1: The query includes the predicted *knows* triples for Trelena and rates them by predicted probability.

Chapter 3

Kernel SUNS

The relationship between kernels and graphs has been the subject of a great deal of academic study. Graph kernels evaluate the similarity between graphs and can be classified into three types: 1) graph kernels based on walks and paths, 2) graph kernels based on limited-size subgraphs, and 3) graph kernels based on subtree patterns [103, 32]. Link prediction on graphs is closely related to semi-supervised learning as surveyed in [109], where the goal is to predict unknown node labels based on known labels in a graph. Kernels for semi-supervised learning have, for example, been defined based on the spectrum of the Graph-Laplacian. Link prediction approaches based on the Gaussian process are presented in [108, 105], and link prediction in relational graphs also has been studied by the relational learning and ILP communities [95, 76, 60]. Kernels for semantically rich domains have been developed by [22] as well.

Most of the kernel approaches discussed in those works cannot easily be applied to the rich semantic domains considered in this thesis. Many approaches have been developed to address a single object type and a single relation type, and the experimental results of the semantic kernels described in [22] are quite limited. In contrast, we have applied our SUNS approach to problems in multi-relational domains in which thousands of potential links are predicted based on hundreds of thousands of features for each of many graph nodes.

The main contributions of this chapter are published in:

1. [42] Yi Huang, Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A scalable kernel approach to learning in semantic graphs with applications to linked data. In *1st Workshop on Mining the Future Internet*, 2010

2. [45] Yi Huang, Volker Tresp, Maximilian Nickel, Achim Rettinger, and Hans-Peter Kriegel. A scalable approach for statistical learning in semantic graphs. *Semantic*

Web Journal, 5(1):5–22, 2014

This chapter is based on the publications above but is rewritten to a large extent. The experiments in [42] were not carried out by me and therefore they are not included in this thesis.

3.1 The Nyström Approximation

We now assume that for any two instances i and j in the population a kernel $k_{i,j}$ is defined. A subset of the population of size n, i.e., the sample, defines the training set D. Let K be the $n \times n$ kernel matrix (i.e., Gram matrix) for the training instances. In many applications, n can be very large, and following [104], we therefore use the Nyström approximation to scale up kernel computations to large data sets.

The Nyström approximation is based on an approximation to eigenfunctions and starts with the eigendecomposition of the kernel matrix

$$K = UDU^\top \tag{3.1}$$

where K is symmetric, U is orthonormal, and D is a diagonal matrix with the descending eigenvalues. The Nyström approximation to the kernel for two arbitrary instances i and j can be written as

$$k_{i,j} \approx k_{.,i}^\top \, U_r \, \mathrm{diag}_r \, (1/d_l) \; U_r^\top \; k_{.,j} \tag{3.2}$$

where $\mathrm{diag}_r \, (1/d_l)$ is a diagonal matrix containing the inverse of the r leading eigenvalues of D and where U_r contains the corresponding r columns of U[1]. Here, $k_{.,i}$ is a column vector of kernels between instance i and the training instances and the same applies also to $k_{.,j}$.

Following this process reveals two special cases of interest. First, the vector of approximate kernels between a statistical unit i and all units in the training data can be written as

$$k_{.,i} \approx U_r U_r^\top k_{.,i} \tag{3.3}$$

and the matrix of approximate kernels between all pairs of units in the training data is

$$K \approx U_r \, \mathrm{diag}_r \, (d_l) \, U_r^\top \tag{3.4}$$

[1]Based on this approximation the rank of any kernel matrix is less than or equal to $r \leq n$.

These modified kernels can now be used in kernel approaches such as SVM learning or Gaussian process learning. On the one hand, if $r \ll n$, the reduced-rank approximation Equation (3.4) can greatly reduce the computational requirements. [2]. On the other hand, using the Nyström m method allows us to select a smaller subset of D to approximate the kernel computations. While the selection strategy used in this method has been the subject of some recent work, it is outside the scope of the thesis. The resulting computational complexity is reduced to $O(m^2 n)$ [104], where n is the number of the instances, and m is the size of the selected subset (usually $r \leq m \ll n$).

3.2 Kernel SUNS

So far, our discussion has been quite general, and the Nyström approximation can be used for any kernel defined between instances in the population. As discussed at the beginning of this chapter, there are several applicable kernels defined for nodes in a graph, but most are not directly applicable to the rich domain of semantic graphs with many different node and relation types. One notable exception is [22], which defines kernels that exploit rich ontological background knowledge.

In [96], we present the kernels that can be used in the SUNS framework. As described in the previous chapter, the random variables represent the likelihood of links where the statistical unit is the subject or the object, while covariates form additional features, e.g., aggregated information. Although those features are explicitly calculated, a kernel approach is nevertheless preferable since, in the applications we are considering, the number of the features can be quite large, whereas the size of the sample n can be controlled more easily.

Following the notations used in Chapter 2, let us assume that we have a training data set $D \subset \mathbb{R}^{n \times (m+l)}$ containing n instances. An instance i consists of m inputs or covariates $x_i = (x_{i,1}, \ldots, x_{i,m})$ and l targets or random variables $y_i = (y_{i,1}, \ldots, y_{i,l})$. Let us also assume that the goal is to learn a model W that is able to predict y_i in the form of $y_i \approx \hat{y}_i = k(\cdot, i)^\top W$ (cf. Equation (2.1)), where W is an $n \times l$ weight matrix and $k(\cdot, i)$ is the kernel vector of the instance.

As described in Section 2.1.5, a regularized least squares cost function can be formulated as

$$\ell(Y, \hat{Y}) = Tr((Y - KW)(Y - KW)^\top) + \lambda Tr(W^\top K W) \tag{3.5}$$

[2] We use the Nyström approximation slightly differently from [104]. There, Equation (3.1) is used on a submatrix of K and Equation (3.4) is then used to approximate K.

where $Tr(\cdot)$ is the trace of a matrix and $Y = (y_1, \ldots, y_n)^\top$ are the targets of the instances in D. If we use the Nyström approximation for the kernel computations, we obtain the least squares solution for the weight matrix.

$$\begin{aligned} \hat{W}_{ls} &= (K^\top K + \lambda K)^{-1} K^\top Y \\ &\approx U_r \operatorname{diag}_r \left(\frac{1}{d_l + \lambda} \right) U_r^\top Y \end{aligned} \tag{3.6}$$

The prediction for the training data (i.e., smoothing or transduction) is carried out

$$\begin{aligned} \hat{Y} &= K \hat{W}_{ls} \\ &\approx U_r \operatorname{diag}_r \left(\frac{d_l}{d_l + \lambda} \right) U_r^\top Y \end{aligned} \tag{3.7}$$

and, in general, a prediction for any instance *not* included in the training set (i.e. induction), can be made

$$\hat{y}_i = k(\cdot, i)^\top \hat{W}_{ls} \tag{3.8}$$

Let us consider some individual kernels. If a kernel is defined as an inner product of the covariates $k^x_{i,j} = \langle \phi(x_i), \phi(x_j) \rangle$, where ϕ is a feature mapping that maps instances from the original feature space into a reproducing kernel Hilbert space, then in the case of the linear kernel, the Nyström approximation is equivalent to the regularized PCA regression in that covariate space. Another kernel is $k^y_{i,j} = \langle \phi(y_i), \phi(y_j) \rangle$ In this case, the approximation is equivalent to the regularized matrix reconstruction obtained by using kernel PCA, which is often used in collaborative filtering. In the latter case, the low-rank Nyström approximation is not only required in order to obtain a scalable solution but also necessary in order to obtain accurate predictions at all: with $\lambda \to 0$ and $r = n$ we would obtain the trivial solutions $\hat{Y} = Y$. Finally, with $k^z_{i,j} = \langle \phi(z_i), \phi(z_j) \rangle$ where $z_k = (\alpha x_k^\top, y_k^\top)^\top$ and $k = \{i, j\}$ we obtain the reduced-rank penalized kernel regression algorithm utilized by the SUNS framework [41]. Here, α is a positive weighting factor balancing the influence of the two information sources.

It is worth noting that, compared to RRPP introduced in Section 2.1.5, the kernel approach defined in this chapter is capable of modeling non-linear dependencies among covariates and random variables. Moreover, although the formulae of the kernel SUNS look similar to those in Section 2.1.5, here, U are the eigenvectors of Gram matrix $K = DD^\top$ and d_l are the eigenvalues of K, i.e., the squares of the singular values s of D. When computing K, the Nyström method allows us to control the size of the selected instances easily.

3.3 Experiments

We evaluated the kernel approach on a data set gathered from DBpedia [4], which is based on information extracted from Wikipedia. The task was to predict the political party of the members of the German Bundestag. The instances in the data set were the German politicians, while the features included German political parties, the politicians' age and birthplace, and other relevant features extracted from textual information. In our experiments, we used the Pearson Correlation on the disambiguated original data as a baseline and investigated the kernel PPRR during more and more information being added. We observed that performance increased as more information was exploited, and that the approach's prediction quality peaked when using all information. Additionally, we discovered that a proper value of the parameter α could improve the performance of the approach. For more details, see [42].

3.3.1 Scalability

In the relational domains we are modeling, the training data X are very sparse, and the reduced-rank matrix reconstruction can be calculated efficiently. Given a sparse random $N \times M$ matrix X, we might first construct the kernel matrix via $K = XX^T$ and then use sparse SVD to obtain U_r. Figure 3.1 shows experimental results of this approach. The graph at top left shows computational time for the SVD as a function of N (dashed red line) and illustrates an approximately linear dependency related to the fact that the number of rows of U is N as well. In this experiment, $r = 50$, $M = 10^6$, and the number of nonzero entries in X is $p = 10^6$. The top right graph shows computation time for the SVD as a function of M (dashed red line). Here, using $p = 10^6$, $N = 10^5$, and $r = 50$, computation time decreases as M increases, because K becomes less dense. The bottom left shows an approximately quadratic dependency of the computational time for the SVD on p ($M = 10^6$, $N = 10^5$, $r = 50$) (dashed red line). Note that the last data point in the plot is a system with $p = 10^7$ that requires only 10 minutes of computation. Finally, the bottom right graph shows the dependency of the SVD's computational time on r ($M = 10^6$, $N = 10^5$, $p = 10^6$) (dashed red line). A tenfold increase in r results in an approximately tenfold increase in computational cost. Each of the four graphs also shows the computational time (blue line) for calculating $K = XX^T$, which is negligible in comparison. A prediction for a new instance (i.e., a new row in X) can efficiently be calculated using Equation (3.8).

It is worth noting that for a sizable matrix with 10^5 rows, 10^6 columns, 10^7 nonzero

elements, and a rank of $r = 50$, the computation takes only approximately 10 minutes on a standard laptop. For kernel matrices where $K = XX^T$ becomes dense, one might employ the Nyström approximation technique described in Section 3.1, which reduces the time complexity to $O(nm^2)$, where n is the number of the rows and $m \ll n$ is the size of the selected subset of the rows.

3.4 Remarks

In this chapter, we discussed kernel SUNS as a general kernel approach using the Nyström approximation. We demonstrated that the scalability of the overall approach can be guaranteed by controlling both the number of instances considered in the kernel computations and the rank of matrix decomposition. In addition, we showed that it is possible to control the number of local features used to derive the kernel. The approach SUNS presented in Chapter 2 is a special case of the kernel SUNS.

We applied the approach to DBpedia, which is a vast and rich semantic knowledge base. At the time of our experiments, it contained approximately 3.4 million concepts, and now (as of the 2016-10 release) consists of 13 billion pieces of information (RDF triples). Both the quantity and the quality of DBpedia continue to improve, and we believe that the general kernel approach can be utilized in many applications and can open additional avenues of research on this knowledge base in the future.

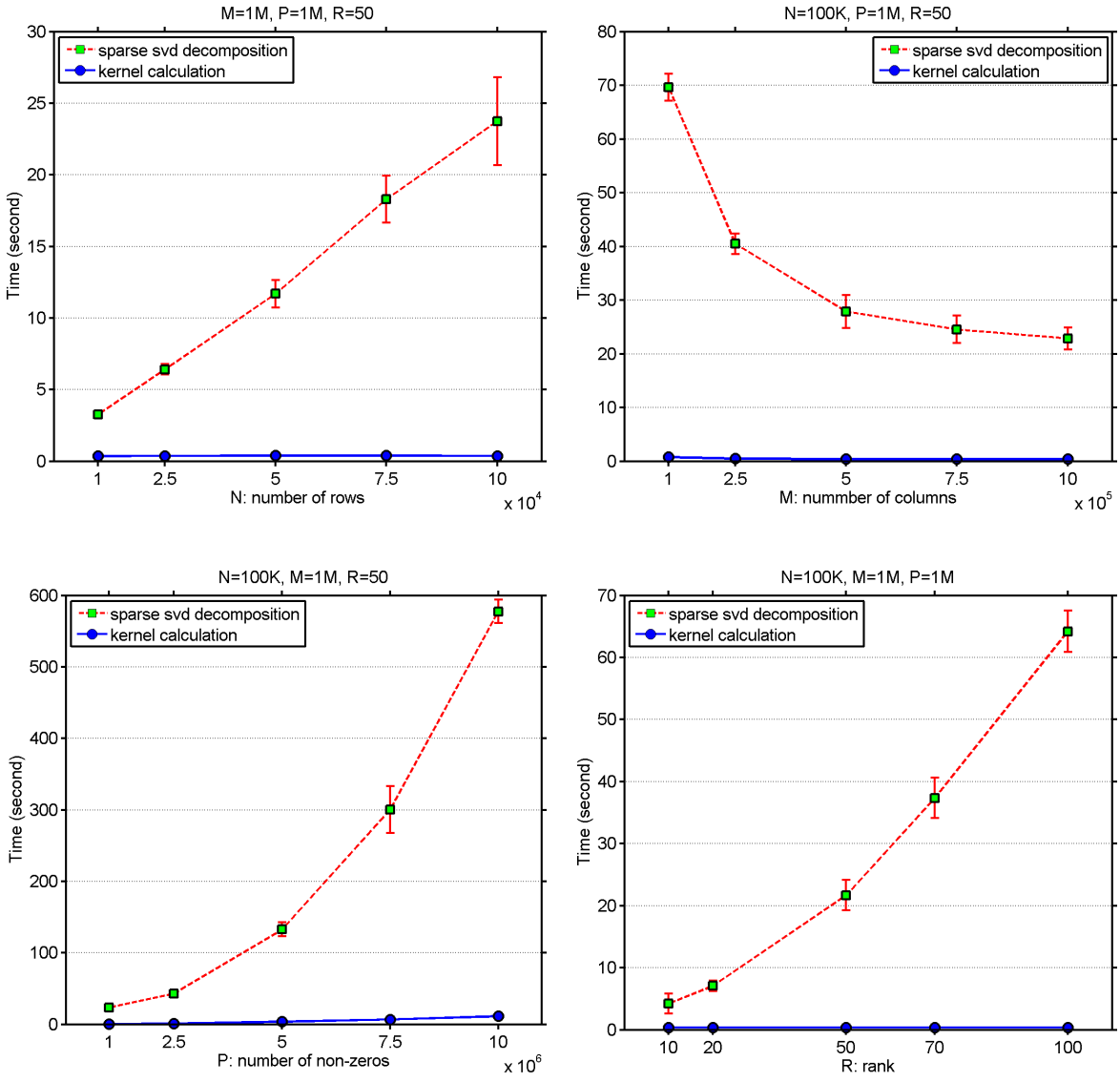


Figure 3.1: The time complexity of the kernel model

Chapter 4

R-Model

In this chapter, we introduce a probabilistic model that relies on a relation-oriented sampling assumption and enables the handling of more complex relationships in a probabilistic manner. Rather than a single entity type (such as a person, patient, or politician), by using this probabilistic model in the SUNS framework, it becomes possible to easily model relationships to which multiple entity types participate and also to model contextual information as necessary. By way of introduction, we describe the traditional object-oriented sampling model discussed in the previous chapters. We then present our relation-oriented sampling model, which we call the R-model, and illustrate it using a concrete example. Finally, we evaluate the R-model's effectiveness.

The main contributions of this chapter are published in:

1. [98] Volker Tresp, Yi Huang, Xueyan Jiang, and Achim Rettinger. Graphical models for relations - modeling relational context. In *KDIR 2011 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, Paris, France, 26-29 October, 2011*, pages 114–120, 2011

In this publication, I co-created the R-model and accomplished all empirical studies. This chapter is based on [98] but is rewritten.

4.1 Object-Oriented Sampling Assumption

Traditionally, statistical units (i.e., data points) are associated with objects, and statistical models describe the statistical dependencies between attributes of those objects. A typical example in the medical domain might involve analyzing the dependencies between the attributes of a patient population by using a Bayesian network. Applying the SUNS

framework to this example, a data matrix can be generated in a methodical way wherein rows are defined by unique identifiers belonging to individual patients and their attributes form the columns. In a case like this, a fundamental task is to predict whether a new patient would come from the same population (density estimation), or what value a variable has to assume to be such that the likelihood that the patient belongs to the same population is maximized (predictive modeling).

This approach is commonly used for modeling relational domains as well. For instance, one might analyze the preferences of a population of U users based on their properties and their known preferences for I items[1], e.g., *buy(User, Item)*. In this scenario, users' preferences are essentially treated as properties of the users as well (Figure 4.1, Left). In [15], authors describe a Bayesian network where each item is represented by a binary node x_i and the state of the node–$(x_i = 1)$ or $(x_i = 0)$–indicates whether a user has purchased a given item. The Bayesian network then estimates a joint probability of all items.

$$\hat{P}(x_1, \ldots, x_I). \tag{4.1}$$

In the SUNS framework, estimating this probability can be viewed as a predictive task: For a user u, the estimated probability is proportional to the variable values predicted by using the kernel approaches in the last chapter, namely $\hat{P}_u(x_1, \ldots, x_I) \propto \hat{y}_u = k^\top(\cdot, u)\hat{W}$ (cf. Equation (3.8)).

These models raise the non-trivial problem of distinguishing between unknown relations and relations that are known not to exist. For example, in the Bayesian networks in [15] and the Dependency Networks in [37], missing relations are treated as not-to-exist, whereas in [57, 106, 27, 33], Gibbs sampling and loopy belief propagation are used for dealing with missing relations as unknown.

4.2 Relation-Oriented Sampling Assumption

Rather than defining an *entity* as a statistical unit, if we define now an observed *relation* as a statistical unit instead–a tuple describing a relation or an event between two objects, for instance (Figure 4.1, Right)–then the population consists of all true events, and any sample taken from that population is a subset of those true events. Thus, whereas we assumed in the previous section that either users or items define the rows in the data matrix, here we assume that each observed instantiated relation defines a row.

[1] U and I are the number of users and the number of items respectively.

	i_1	i_2	i_3	i_4	...
u_1	*1*	*1*	*0*	*1*	...
u_2	*1*	*0*	*0*	*1*	...
...	...	...	...	...	...

	User	Item
id_1	u_1	i_1
id_2	u_1	i_2
id_3	u_2	i_4
...	...	...

Figure 4.1: Left: In traditional statistical point of view, each row is defined by a user, the columns represent various items, and a *one* indicates that a user has purchased a given item. Right: Each row is defined by an event user-buys-item, which is the sampling assumption introduced in this thesis.

In the relationship *buy(User, Item)*, which has two attributes $User$ and $Item$, a relation-oriented sampling assumption would lead to a data matrix containing two columns encoding the user and the item respectively, and the model below would estimate the probability of a *buy* relation between a particular user u and a particular item i.

$$\hat{P}(a_{user} = u, a_{item} = i). \tag{4.2}$$

Note that whereas Equation (4.1) estimates a joint probability distribution over I binary variables, this equation describes a multinomial probabilistic model with two variables $User$ and $Item$ which have U and I states respectively.

Given that the relationship modeled using this equation is now generalized from two to $2 < A \in \mathbb{N}$ attributes, all attributes are now informative with respect to determining the existence of a relation. The equation needs to extend to evaluate $P(a_1, \ldots, a_A)$, i.e., the probability that a new relation instance with attributes $a_1, \ldots, a_A$ is likely to exist. Alternatively, it might be interesting to predict the most likely value of one of the attributes given other attributes, such as $P(a_1|a_2, \ldots, a_A)$. For instance, the probability of an item a_1 given a user a_2 and given contextual information $a_3, \ldots, a_A$, meaning the probability of a user a_2 having purchased an item a_1 in contexts of $a_3, \ldots, a_A$.

In object-to-object relationships, variables typically contain many states and a contingency table involving all variables may be very sparse, i.e., may include a large number of zeros or values near to zero. Graphical models have been proven to be quite effective at handling high-dimensional data [63], and we, therefore, apply them in this thesis as well. As discussed earlier, we apply graphical models in a relation-oriented SUNS framework where relations constitute instances and where we are concerned with one relationship instead of a whole network of entity types and their relationships. For our purposes, Bayesian

networks and decomposable models are most suitable for defining such a framework. In a Bayesian network, the probability distribution factors as

$$\begin{aligned} P(a_1, \ldots, a_A) &= \prod_{i=1}^{A} P(a_i | \mathbf{par}(a_i)) \\ &= \prod_{i=1}^{A} \frac{P(a_i, \mathbf{par}(a_i))}{P(\mathbf{par}(a_i))}. \end{aligned} \tag{4.3}$$

Typically a Bayesian network is depicted as a directed graphical model without directed loops. In this model, $\mathbf{par}(a_i)$ denotes the direct parents of a_i.

Given a Bayesian network structure, the task ofEquation (4.3) is to model $P(a_i|\mathbf{par}(a_i))$, or equivalently, $P(a_i, \mathbf{par}(a_i))$ for each attribute a_i. In cases where variables have many possible states, matrix and tensor completion methods have been successful at dealing with missing values in the past, and we also make use of those methods in our R-model as described in the following section.

4.3 An Example Illustrating R-Model

4.3.1 A Social Network

GetGlue (http://getglue.com) [2] is a social network where users connect with one another and share Web navigation experiences. GetGlue uses semantic recognition techniques to identify entities such as books, movies, and other similar topics and publishes these entities in the form of data streams. Users can read the streams and receive recommendations on an exciting discussion of entities from their friends. Both the social network's data and the real-time streams it generates are accessible via Web APIs. Users and their relationships with others are described by well-known Semantic Web terminologies, such as user names and the *knows* relationship defined by Friend of a Friend (FOAF) terms, and the *follows* relationship is recognized by the Semantically Interlinked Online Communities (SIOC) [3]. Objects in the social network data represent real-world entities with a name and a category, such as movies or books, and resources represent information sources describing those objects, such as web pages about a particular movie or book.

We explain R-Model based on an example: recommending movies to users in GetGlue. The recommendation task is essentially a probability density estimation problem since we estimate the probability that an unknown user-movie pair belongs to the population.

[2] GetClue changed its name to tvtag in 2014, and one year later, it was dissolved.

[3] http://sioc-project.org/

4.3.2 Modeling User-Movie Events

For recommending movies to users in GetGlue, we model the event that a user watches a movie, and that we call user-movie events in this section. A graphical model consists of two attributes: the user and the movie (Figure 4.2). Known user-movie events define the rows in the data matrix, and the columns consist of two variables with as many states as the number of users and movies, respectively. A contingency table C is then constructed where the two categorical variables are *user* and *movie*. Each entry in the table $c_{u,m}$ counts how often user u has watched movie m and represents a random sampled event from the population. By dividing the entries by the overall count of the user-movie events, we can interpret the entries as estimates for the probabilities of observing user-movie pairs under this sampling assumption, i.e., as a maximum likelihood estimate of $P(u, m)$. Since this matrix will contain many zero entries, the maximum likelihood estimates are notoriously unreliable. Following common practice, we smooth the matrix using a matrix factorization approach and perform an eigenvalue decomposition of the Gram matrix of the contingency table $CC^T = UDU^T$ to obtain a low-rank approximation of C [44]

$$\hat{C} = U_r \operatorname{diag}_r \left(\frac{d_l}{d_l + \lambda} \right) U_r^\top C \tag{4.4}$$

where $\operatorname{diag}_r \left(\frac{d_l}{d_l+\lambda} \right)$ is a diagonal matrix containing the r leading eigenvalues in C and where U_r contains the corresponding r columns of U. λ is a regularization parameter. After proper normalization of $\hat{C}$, the entries can be interpreted as $\hat{P}(u, m)$, i.e., an estimate of the probability of observing the relation that user u watches movie m. [4] Recommendations for users can now be derived from $\hat{P}(u, m)$.

It is worth noting that matrix completion is an active area of research, and many other matrix completion methods are applicable to this model as well.

Compared to Equation (3.7) in the previous chapter, the target variables Y in Equation (3.7) are replaced by the (normalized) contingency table C. The former contains binary entries indicating whether the state of a variable is true or false, while the latter contains ordinal entries signifying the probability that an event is sampled from the population. In recommendation systems, such a probabilistic model is intuitively more suitable, since user preferences are usually represented as ratings and can trigger events like watch-movie and buy-good repeatedly.

[4] The normalization takes care that all entries are non-zero and are smaller than one. Incidentally, this step turns out to be unnecessary in the regularized reconstruction, since after matrix completion, all entries already obeyed these constraints. A second step ensures that the sum over matrix entries is equal

Figure 4.2: A graphical model for the dependencies between users U and movies M.

4.3.3 Adding Last Movie Watched

Indeed, the user-watches-movie process involves a sequential character that the model in Figure 4.2 so far cannot capture. If we consider the last movie that a user has watched as additional information [87], we then obtain a truly ternary relationship *watches(u,m,l)* consisting of user, movie and last movie l watched by the user. The approach to modeling *watches(u,m,l)* followed in [87] considers a three-way contingency table and applies tensor factorization as a means of tensor smoothing. [87] argues additionally that general tensor factorization, such as PARAFAC or Tucker [56], are too difficult to apply in such a situation where the contingency table is very sparse and suggests that a simplified additive model is applied instead. In our approach, we propose that a graphical model, which we illustrate in Figure 4.3 (left). [5] The model indicates that 1) the last movie watched by a user directly influences the next movie that the user watches, and 2) given the movie, the last movie watched and the user are independent. The advantage lies in that we do not need to re-adapt the user-movie model but can independently model the movie-last-movie dependency. Our model calculates empirical probabilities of movie-last-movie pairs based on the contingency table, smooths the table using matrix factorization, and obtains $\hat{P}(m,l)$. We then combine both models in the form of

$$\begin{aligned}\hat{P}(u,m,l) &= \frac{\hat{P}(u,\mathbf{par}(u))}{\hat{P}(\mathbf{par}(u))} \cdot \frac{\hat{P}(m,\mathbf{par}(m))}{\hat{P}(\mathbf{par}(m))} \cdot \frac{\hat{P}(l,\mathbf{par}(l))}{\hat{P}(\mathbf{par}(l))} \\ &= \frac{\hat{P}(u,m)}{\hat{P}(m)} \cdot \frac{\hat{P}(m,l)}{\hat{P}(l)} \cdot \hat{P}(l) \\ &= \frac{\hat{P}(u,m)\hat{P}(m,l)}{\hat{P}(m)}\end{aligned} \tag{4.5}$$

In contrast to [87], we obtain a *product* of local models rather than a *sum*.

to one.

[5] A link from the last movie to the movie might appear more plausible. If one changes the direction of this link, the link between user and movie would need to point from movie to user, such that no collider (more than one link pointing to the same node) appears. With a collider one would need to use a tensor model as a local model.

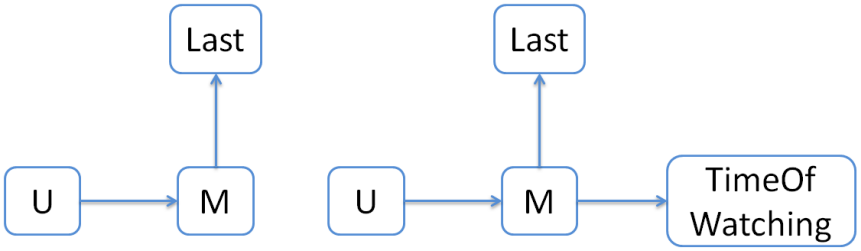


Figure 4.3: Left: As additional information, the last movie, which the user has watched, is added. Right: The month when the user watches the movie is added.

4.3.4 Adding Time of the Event

As we know, movies' popularity changes over time, and a movie rated highly by audiences during one timeframe may decline in popularity later. Consequently, information about when users watched a movie (t, or time when the movie was seen) may be influential in determining which movies other users are likely to see[6]. The graphical model in Figure 4.3 (left) is extended by adding the time of watching in units of the month to handle with the quaternary relationship *watches(u,m,l,t)*. The graphical model is sketched in Figure 4.3 (right). We form an empirical estimate based on the month-of-watching contingency table and obtain $\hat{P}(m,t)$, then combine the three models as follows.

$$\hat{P}(u,m,l,t) = \frac{\hat{P}(u,m)\hat{P}(m,l)\hat{P}(m,t)}{(\hat{P}(m))^2} \tag{4.6}$$

4.4 Empirical Study 1

For evaluation, we gathered 9707 movies watched by 3076 users in a timeframe of 44 months, resulting in a user-movie contingency table and a last-movie-movie contingency table with sparsities of 1.8% non-zero entries and 1.21% non-zero entries, respectively. For each user in our sample, we randomly withheld one movie the user reported having watched in order to test and evaluate the predictive ability of the following models:

Friendship: A baseline model that recommends to a user movies that her friends have watched. This scenario is typical in social networks and operates according to the assumption that friends are likely to have common interests and like to share their experiences with friends.

[6] Also, a movie can only be watched after its release.

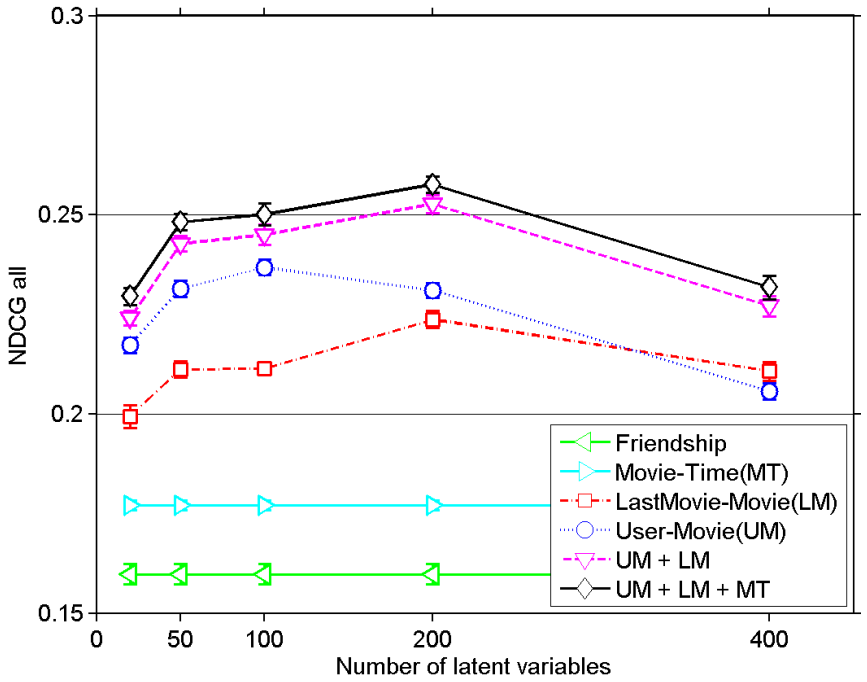


Figure 4.4: Experimental results on GetGlue data without regularization. Models' NDCG scores are plotted as a function of the rank r in the matrix completion.

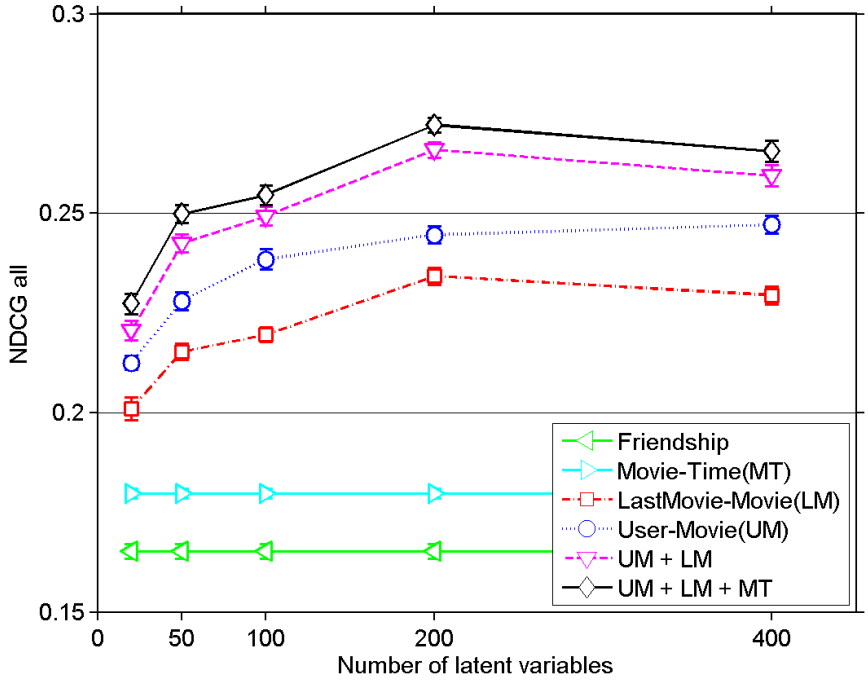


Figure 4.5: The results of the same models as in Figure 4.4, but with the regularization ($\lambda > 0$).

Movie-Time (MT): Another baseline model that recommends movies based only on the time (defined as a month) at which the movie was seen by other users. The same recommendations are made for every user in each particular month. These recommendations can be viewed as indications of which movies were most popular recently.

LastMovie-Movie (LM)): A model capturing only the dependency between the last movie a user watched and the next one to watch. This sequential effect can be observed in many movie series such as Star Wars and The Avengers.

User-Movie (UM)): The basic R-model estimating $\hat{P}(u, m)$.

UM+LM: The combined R-model estimating $\hat{P}(u, m, l)$.

UM+LM+MT: The combined R-model estimating $\hat{P}(u, m, l, t)$.

In Figure 4.4 and Figure 4.5 we plot NDCG scores [48] (described in Appendix 6) as a function of the rank r in the matrix approximation. All experiments were run with 5-fold cross-validation such that an error bar can be displayed. Figure 4.4 shows the results of the models without regularization ($\lambda = 0$), while Figure 4.5 presents the results of the regularized models ($\lambda > 0$). The baseline *Friendship* performed worst. *MT* was more effective, but worked only in cases where movie recommendations matched the overall popularity of a movie in a given month. The performance of both baselines was independent of the regularization.

By contrast, the regularized models showed much better performance. *LM* recommended movies based on information about the movies that users watched most recently. This model addressed only the Markov chain property of the event of watching movies. Because *LM* performed worse than *UM*, which is based on the classical user-movie model, we can conclude that user preferences are more informative than sequential information in the data. When combining both sources of information in *UM + LM*, predictive performance improved markedly. Finally, the superior performance achieved by combining all three models in *UM + LM + MT* in addition to the months during which users watched various movies confirms the benefits of the R-model.

We also investigated how a simple average of the probabilities of the individual models would perform. Figure 4.6 shows that combining the models using a simple average did slightly improve the quality of recommendations, but the improvement was not comparable with the results obtained by using R-model at all.

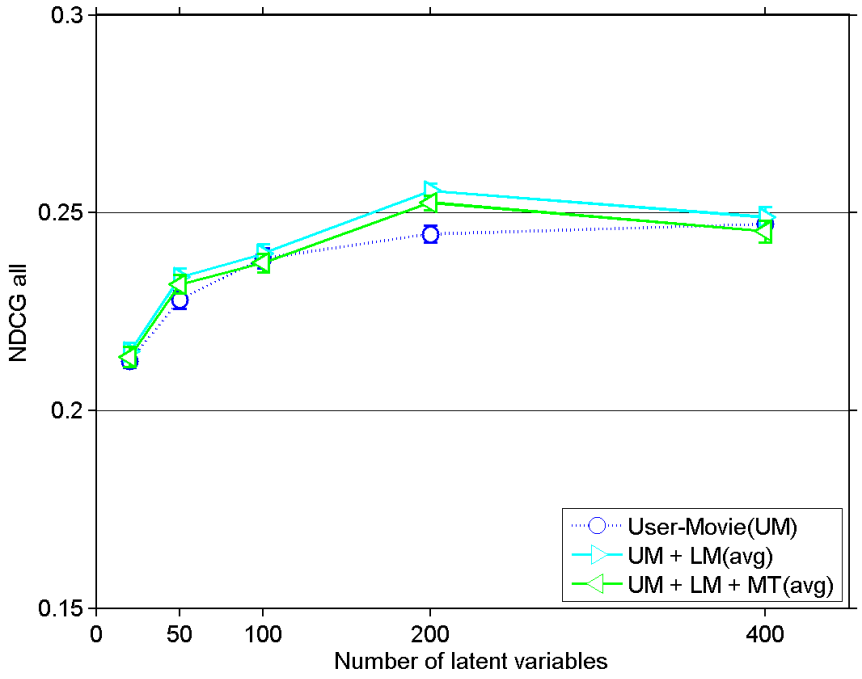


Figure 4.6: The figure shows the results obtained by simply averaging the estimated probabilities generated by each individual models.

4.5 Empirical Study 2

Based on the same data set, we tested the R-model in two additional scenarios. In the first or the *most recent movie scenario*, the movie a user watched most recently withdrew and assumed to be unknown. In the testing, we judged how likely this movie is recommended to watch next. This scenario corresponds to a traditional approach to recommending movies to watch next based on the series of movies a user watched before. In the second or the *random movie scenario*, we randomly dropped one movie out and used it as test data for each user, exactly like we have done in the first empirical study. This setting corresponds more closely to traditional collaborative filtering tasks, where contextual information, meaning here sequential effect, is not considered. In both scenarios, movies a user already watched are not recommended again to the same user. Repeated recommendations might be handled differently in other applications, as in e-commerce applications where might reasonably wish to buy the same item(s) more than one time. In the *random movie scenario*, we repeated the experiment 5 times such that error bars can be produced (see figures below).

4.5.1 Methodology

We introduce an additional baseline *most popular*.

most popular: A baseline model recommending the same list of movies to every user. The list presents movies in descending order of popularity, with the most frequently watched appearing at the top.

In the experiments of the second empirical study, we used two methods to evaluate the quality of recommendations. First, a model generated a recommended list of k items for each user. If the test item–the most recently watched movie in the most recent movie scenario and a random movie in the random movie scenario–appears in the recommended list, then for that user, we consider the recommendation successful. The ratio of successful recommendations to total recommendations is called the *HitRatio*. Mathematically, the HitRatio is defined as

$$\text{HitRatio}(k) = \frac{1}{|U|} \sum_{u \in U} \delta(T_u \in R_k),$$

where δ is the indicator function, T_u is the user's test item, R_k are the top k recommended items which in our case are the *last movie watched* and the *month of watching*.

Secondly, we applied normalized discounted cumulative gain (nDCG) to evaluate the quality of the recommendations (described in Appendix 6).

Each of these evaluation measures focuses on different aspects of the ranking of recommendations. NDCG emphasizes the top positions in the list of ranked items. Due to the log function on the position k, when the position of a test item is lifted from the second position to the first one, its nDCG score increases much more than when its position is raised from 100 to 99. In contrast, HitRatio gives equal weight to all positions in the ranking list, although it can be made to focus only on the top k items in the ranking list by specifying k.

4.5.2 Results

In this section, we report the results of the models *UM*, *UM + LM + MT* and the baseline *most popular*.

Figure 4.7 (a) and (b) show the HitRatio values for the top 10 movies recommended by the models in the *newest movie scenario* and the *random movie scenario* respectively. We ran each of the models with factorization dimensions $10, 50, 100, 200, 400$ and 1000. Again, for the *random movie scenario* we also report the standard error in Figure 4.7 (b).

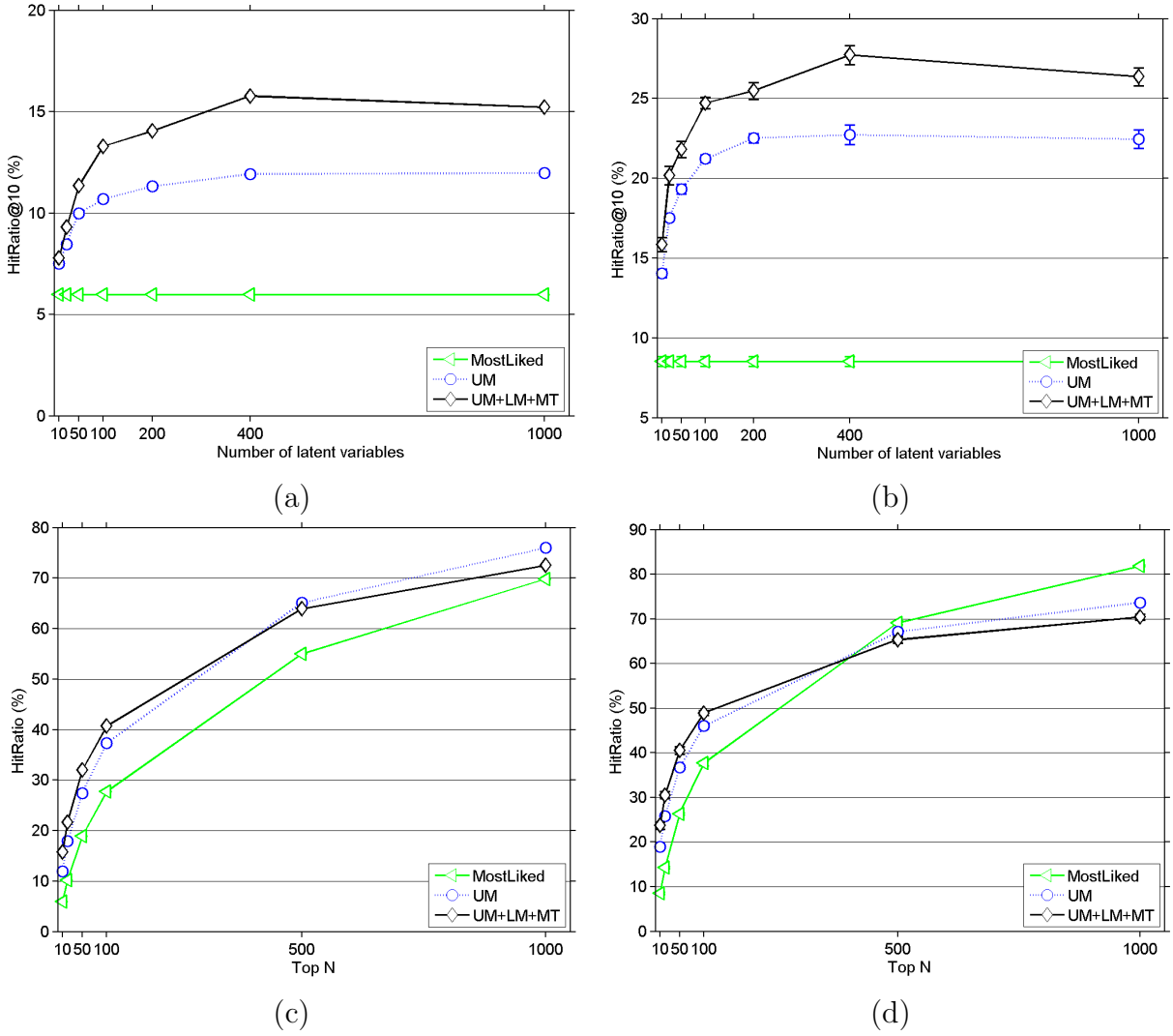


Figure 4.7: Top: (a) and (b) report the HitRatio for a recommended list of top $k = 10$ movies vs. the factorization dimension. Bottom: (c) and (d) report the HitRatio for different values of k with the factorization dimension fixed at 400. Left: (a) and (c) plot HitRatio values in the *newest movie setting*. Right: (b) and (d) plot HitRatio values in the *random movie setting* with error bars showing the standard error.

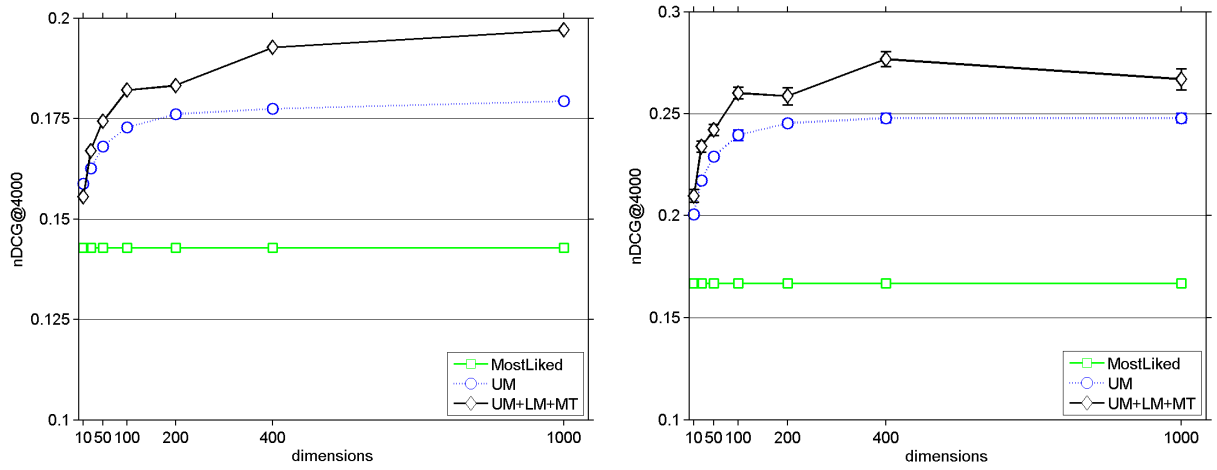


Figure 4.8: The nDCG scores at top 4000 recommended movies for different factorization dimensions. Left: The *random movie setting*. Right: The *newest movie setting*.

These figures show that the combined model *UM+LM+MT* was significantly better than *UM* after the *month of watching* and sequence contextual information was taken into account. In particular, it performed well in the *random movie scenario*, where it correctly recommended the test movie to nearly 28% of the users at $r = 400$.

Figure 4.7 (c) and (d) show the HitRatio values of the tested models against different values of k with fixed r equal to 400. The figures show that for small values of k up to 100 *UM+LM+MT* was the most effective model, followed by *UM*. Neither model produced results comparable to those of the baseline. For large values of k (500 and 1000), we observed that *UM*'s performance reached and then overtook that of *UM + LM + MT*. In the random movie scenario, the HitRatio values of the *most popular* baseline became the most accurate model when k was larger than 500. This phenomenon indicates that many users in the data set prefer the most popular movies to a certain extent, despite their other preferences.

Figure 4.8 Left and Right show nDCG scores achieved in the *newest movie scenario* and the *random movie scenario* respectively. The results are again plotted against values of r from 10 to 1000. The *most popular* baseline generated the worst scores, whereas the R-model *UM* performed much better, and the combined model with all contextual information produced the overall best results.

4.6 Remarks

In this chapter, we described a novel probabilistic model, or R-model, in which we define statistical units by object-to-object relationships. We applied the model to social network data with the goal of recommending movies to users, and we explained how contextual information could be integrated into the R-model in a probabilistic way in order to improve the quality of recommendations. Our work demonstrated the advantages of the R-model's modularity, which allows us to model domains with complex relationships and many variables, including information (such as the last movie watched by a user) that would be difficult or impossible to encode by most other relational learning approaches.

The work we described in this chapter could be extended in at least two ways. First, instead of the regularized matrix factorization we used for approximating the local probability distributions, any other available matrix completion approach could be used as well [17], for instance, the kernel RRPP approach described in the last section, if the number of entities in the training data grows beyond a few thousand. Secondly, in cases where local interactions between more than two many-state variables need to be modeled, one could employ the tensor factorization technique in [56] for the local models.

Based on the experimental results of our empirical studies, several observations can be made. First, the basic R-model *UM* significantly outperformed the baseline models, including the *most popular movies* baseline approach in the second empirical study as well as recommendations based on the *friend-of-a-friend* relationship in the first empirical study. Second, modeling contextual information greatly improved the quality of recommendations, when compared to approaches that considered only the *user-movie* relationship. This result confirms the main advantage of the model presented in this chapter.

Third, the R-model is particularly suitable for recommendation tasks that emphasize the quality of the top-ranked items. Its advantage stems from the matrix factorization technique, which tends to model more popular items and such users who watched an enormous amount of movies and can be viewed as opinion leaders. Both items' popularity and opinion leaders strongly affect the behavior of the whole community in social networks and recommendation systems. Forth, the R-model is robust and insensitive to the number of latent variables. The property can be explained by the fact that, in contrast to other matrix factorization approaches such as SVD, the R-model is regularized. As a result, the R-model easy to use, even by people without machine learning expertise. Fifth, the R-model is efficient and capable of scaling up to large data sets, as illustrated by the fact that we successfully carried out all of our experiments on a standard laptop. For additional discussion of the issue of the scalability, see Section 3.3.1.

Chapter 5

Applications

This chapter describes the application of the SUNS approach to three real-world use cases: social media analysis, disease gene prioritization in life sciences, and a location-based personalized recommendation engine.

The main contributions of this chapter are published in:

1. [7] Davide Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Yi Huang, Volker Tresp, Achim Rettinger, and Hendrik Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intelligent Systems*, 99, 2010

2. [5] Marco Balduini, Irene Celino, Daniele Dell'Aglio, Emanuele Della Valle, Yi Huang, Tony Kyung-il Lee, Seon-Ho Kim, and Volker Tresp. BOTTARI: an augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. *Journal of Web Semantics*, 16:33–41, 2012

3. [6] Marco Balduini, Irene Celino, Daniele Dell'Aglio, Emanuele Della Valle, Yi Huang, Tony Kyung-il Lee, Seon-Ho Kim, and Volker Tresp. Reality mining on micropost streams - deductive and inductive reasoning for personalized and location-based recommendations. *Semantic Web Journal*, 5(5):341–356, 2014

4. [45] Yi Huang, Volker Tresp, Maximilian Nickel, Achim Rettinger, and Hans-Peter Kriegel. A scalable approach for statistical learning in semantic graphs. *Semantic Web Journal*, 5(1):5–22, 2014

These publications are joint works, mainly between Siemens and CEFRIEL, for which I was the main contributor from the side of Siemens. In these papers, I wrote the sections

related to inductive stream reasoning and conducted the associated experiments. This chapter is based on these publications but is mostly rewritten.

5.1 Stream Reasoning for Semantic Social Media Analysis

In social networks, users publish data (often in the form of a stream) from nearly any place and at any time. In combination with the rich background knowledge offered by social networks, the availability of data published continuously by users enables real-time reasoning tasks that produce valuable information for diverse purposes. Determining the hottest topics discussed on Twitter right now or the movies one's friends are likely to watch next on Netflix are two examples. To perform reasoning tasks like those, one needs to connect data stream processing technologies with reasoning methods.

Stream processing has been studied by both the database and data mining communities. Data Stream Management Systems (DSMS) are available on the market, and DSMS features appear in major database products, such as Oracle and DB2. DSMS are designed to process real-time parallel queries of potentially bursty data, but they cannot perform reasoning tasks as complex as those mentioned above.

Online stream mining has been applied in many contexts and for different purposes, including intrusion detection in computer network traffic, recommendation generation in Web searches, and automated real-time decision making using sensor data. These applications represent a paradigm shift in information processing techniques, since data streams are processed on the fly without being stored, and processing units produce their results without explicit invocation. However, little work has been done on applying machine learning to data streams as rich and structured as those we have considered here.

In this section, we briefly describe Stream Reasoning. Then, we introduce an advanced reasoning approach that combines deductive and inductive stream reasoning and uses SUNS as an inductive stream reasoning method, applying this approach to semantic social network data. Finally, we discuss the results of this application.

5.1.1 Stream Reasoning

Introduced in [26], Stream Reasoning is a technique that enables data streams to be merged with rich background knowledge. Stream Reasoning addresses a well-known challenge of extending reasoning methods to support changing knowledge. The reasoning methods can

be categorized into two classes: deductive reasoning and inductive reasoning. Deductive approaches to stream reasoning utilize various methods to revise beliefs based on recent information, while inductive approaches support online data analysis, based on extensive research in data mining and machine learning.

The combination of deductive and inductive stream reasoning extends the notion of stream reasoning, which involves supporting numerous concurrent decision processes by reasoning in real time on large and potentially noisy data streams. Stream Reasoning involves three key concepts related to stream processing:

Streams: Data streams are unbounded sequences of time-varying data elements that form a continuous flow of information. Recent elements are more relevant than old ones because they describe the current state of a dynamic system. Because RDF is the data interchange format for reasoners, RDF streams are typically the fuel for stream reasoning. We define RDF streams as ordered sequences of pairs, composed of RDF triples and their timestamps T_i:

$$[\ldots, (\langle s_i, p_i, o_i\rangle, T_i), (\langle s_{i+1}, p_{i+1}, o_{i+1}\rangle, T_{i+1}), \ldots]$$

Timestamps are annotations of RDF triples. They are monotonically nondecreasing in the stream ($T_i \leq T_{i+1}$), and adjacent triples can have the same timestamp if they occur at the same time.

Windows: Traditional reasoning problems assume that all available information should be considered when trying to solve a problem. By contrast, stream reasoning restricts processing to a specific window of concern, focusing on a subset of recent statements in the stream, and ignoring previous statements. However, the cumulative effect of windows processed in the past and present windows can be taken into account.

Continuous processing: The reasoning tasks performed by traditional reasoning approaches have well-defined beginnings and endings corresponding to the times at which the reasoner is given a task and delivers results, respectively. Stream reasoning instead operates according to a continuous model, where tasks are registered and continuously evaluated against streaming data.

In the rest of this section, we focus primarily on inductive stream reasoning and on the following challenges faced by that approach:

- processing a large amount of information in a given time window,

- the structured multi-relational nature of data,
- the sparsity of typically high-dimensional data, and
- the fact that data are often incomplete.

Figure 5.1 shows the architecture of a stream reasoner consisting of a set of specialized components. The *selector* component extracts relevant data from input stream sources by exploiting DSMS's window-processing functionality. Content from a given window is then fed into the next component *abstracter* component, which transforms fine-grained data streams into aggregated events and produces RDF streams as output. The deductive reasoner consists of a SPARQL [1] engine in which C-SPARQL [2] queries are directly registered. The results of this step can be used immediately, or they can be further exploited by two additional workstreams, both consisting of an abstracter and an inductive reasoner. One workstream captures long-term patterns, such as common interests of users or correlations among events, while the other models short-term trends, similar to the hype cycle. The Hype Matrix is populated with the content of the current window, whereas the long-term matrix is relatively stable and updated progressively over time. The two inductive reasoners infer new statements with probabilities, which can be queried using an extended version of SPARQL, as we demonstrated at the end of Chapter 2. This setup can be extended by arbitrarily combining and iterating the deductive and inductive reasoners. For example, it might be helpful to feed the inferences of the inductive reasoner back to the deductive reasoner to deduce further knowledge.

5.1.2 Empirical Study

Experimental Data

We applied the stream reasoner whose architecture we described in the last section to data from the social network GetGlue (http://getglue.com). Figure 5.2 outlines the entities and relationships in the data used in our experiments. The area shaped in turquoise represents background knowledge (described in Section 4.3) where users are connected via *knows* and *follows* relationships, and resources describe objects in the real world, e.g., microblogs about movies. We assume that background knowledge is stable over a relatively long period in comparison with the size of a window, but we allow updates to that stable information

[1] SPARQL stands for SPARQL Protocol and RDF Query Language.

[2] Continuous SPARQL (C-SPARQL) is a SPARQL extension for expressing continuous queries over RDF graphs and RDF streams. [8]

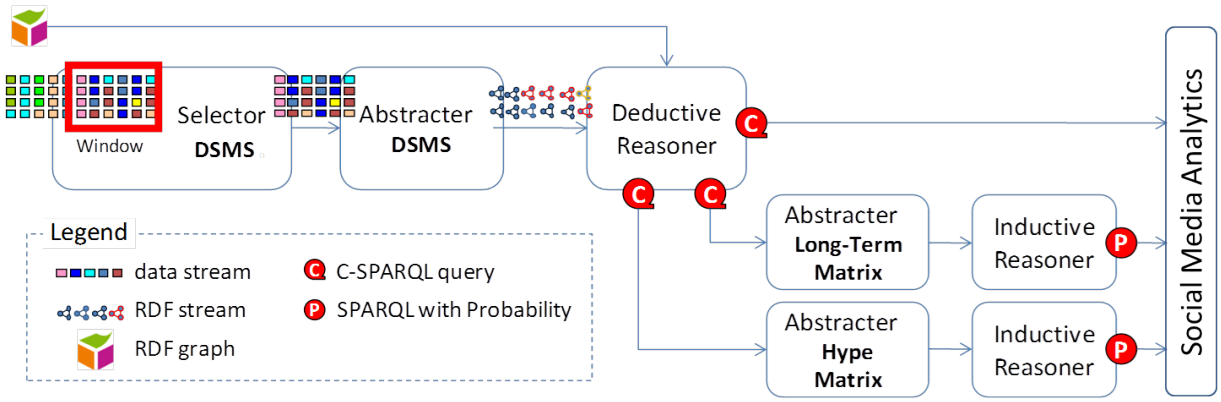


Figure 5.1: Architecture of a stream reasoner as a set of specialized components. We applied deductive and inductive stream reasoner to social media analysis.

so long as they do not interfere with window processing. The area shaped yellow in the figure represents available streaming information that records interactions between users and resources (transitively, between users and objects). The *accesses*, *likes*, and *dislikes* relationships represent interactions occurring when users access[3] resources and express their opinions about them, respectively. We refer to the vocabulary of these relationships with the prefix *sd*. Each interaction of a user U with a resource R generates a triple of the form $\langle U, sd{:}accesses, R\rangle$. A subset of selected interactions expressing opinions generates triples of the form $\langle U, sd{:}likes, R\rangle$ or $\langle U, sd{:}dislikes, R\rangle$. List 5.1 shows examples of possible triples.

```
1 (<:Giulia, sd:accesses, :Avatar>, 2010-02-12T13:18:05)
2 (<:John, sd:accesses, :Twilight>, 2010-02-12T13:36:23)
3 (<:Giulia, sd:likes, :Avatar>,    2010-02-12T13:42:07)
```

Listing 5.1: Examples of triples generated when users interact with resources.

To gather a data set for the evaluation, we performed a predefined C-SPARQL query on GetGlue data to extract RDF streams whose timestamps fall between 19 February and 22 April 2010. We then selected 245,860 interactions generated by 2,457 users. Finally, we applied the SUNS framework to transcode the RDF streams into a data matrix. The transformation process is described in the next section. In particular, we examined the interactions of the "user likes movie" relationship. This segment of the data matrix was

[3]An access can refer to different events such as create, read, and retweet, including likes and dislikes.

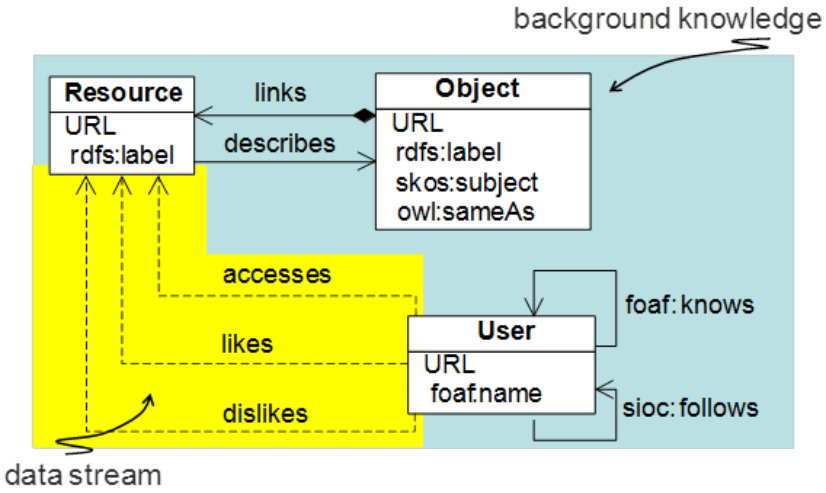


Figure 5.2: Entities and relationships in our experiments. The UML diagram shows that objects represent real-world entities and resources represent information.

extremely sparse, with only 0.002 percent non-zero elements. To make our evaluations statistically significant, we removed users with few interactions and items evaluated less than five times. After that pruning step, the resulting sub-matrix consisted of 1,455 users and 7,724 features, with a sparsity of 0.02 percent. The items with the most user ratings were 2,467 "liked movies", followed by 1,378 "liked music" items, 1.241 "liked recording artists", 592 "liked movie stars", 592 "liked T.V. shows", and 579 "liked video games". The remaining 18 items in the sub-matrix were each mentioned less than 250 times.

Inductive Reasoners

We applied the SUNS framework to the semantic data streams in the yellow area in Figure 5.2 as an inductive reasoner first by defining the statistical unit, population, sampling procedure, and features of the population. A statistical unit is an object of a certain type, such as a user. A population is the set of statistical units under consideration. In our experiments, we defined Glue social network users as the population. We sampled a subset of the users to train models. Then, based on the sample, SUNS constructed data matrices by associating RDF triples with corresponding statistical units and transforming them into the matrices (details in Section 2.1). The sampled users define the rows in the matrices, and their features derived from the associated RDF graph define the columns, such as movies and music they have liked. The entries in the matrices are binary and refer

to the existence of triples. An entry is *one* when the corresponding triple exists and *zero* otherwise. Given rows representing users and columns representing movies that the users like, for instance, a *one* in the (i, j) entry indicates that the i-th user has rated the j-th movie as liked. Otherwise, the user did not rate that movie, and thereby it is unknown whether the user likes it.

After data matrices were constructed, we evaluated two multivariate machine learning models: Singular Value Decomposition (SVD) and the RRPP model introduced in Section 2.1.5. The approaches estimated unknown matrix entries via a low-rank matrix approximation. Note that RRPP can be implemented through regularized low-rank SVD. The matrices were reconstructed by replacing *zero* entries with the estimated probabilities that the corresponding triples would be true.

In the following example, an ad-hoc query seeks information on which movies the user Guilia would probably like, even if she has not seen them yet, and the system uses the most recent window in the stream to predict the probability that she would like those movies (see List 5.2).

```
1 SELECT ?movie ?prob FROM STREAM
2 <http://streamingsocialdata.org/interactions>
3         [RANGE 30m STEP 5m]
4 WHERE { :Giulia sd:likes ?movie . WITH PROB ?prob
5         ?movie a yago_Movie .
6         FILTER ( ?prob > 0 && ?prob < 1 )
7 } ORDER BY DESC(?prob)
```

Listing 5.2: A C-SPARQL query

```
1 (:WutheringHeightsTvMovie, 0.8347)
2 (:StarWars, 0.5693)
```

Listing 5.3: The results of the query

At line 4 in List 5.2, the keyword *WITH PROB* extends SPARQL by enabling it to query statements with probabilities. The variable *?prob* assumes the value *one* for the movies she has liked and values between *zero* and *one* representing the estimated probability of movies that she might like and probably watch next. The clause *ORDER BY* sorts movies by their probabilities in decreasing order. The results are a sorted list of pairs consisting of a movie title and predicted likelihood (see List 5.3).

5.1.3 Evaluation

To demonstrate the effectiveness of stream reasoning for social media analysis, we used the accuracy of top-N movie recommendations as the evaluation metric. The evaluation consisted of two parts. First, we compared diverse inductive reasoning approaches with conventional recommendation methods, some of which were performed by deductive stream reasoning. In this part, we applied four conventional recommendation methods as baselines:

MostLiked: a method that recommends a list of the overall most-liked movies to all users; the list is generated by running a registered C-SPARQL query (see List 5.4).

FriendLiked: a method that recommends movies to users based on the most-liked movies of their friends, which are also obtained by executing a registered C-SPARQL query (not shown).

UserKNN: a method that delivers recommendations using k-nearest neighbor (kNN) regression based on the cosine similarity of users [66].

ItemKNN: a method that proposes recommendations using k-nearest neighbor (kNN) regression based on the cosine similarity of items.

```
1 REGISTER STREAM MostLiked COMPUTED EVERY 1d AS
2 SELECT ?movie (COUNT(?user) AS ?noOfUser)
3 FROM STREAM <http://streamingsocialdata.org/interactions>
4         [RANGE XX STEP XX]
5 WHERE { ?movie a yago_Movie .
6         ?user sd:likes ?movie .
7 }
8 GROUP BY ?movie
9 ORDER BY DESC(?nrOfUser)
```

Listing 5.4: A simple registered C-SPARQL query. This query returns a global list of liked movies.

We carefully tuned the parameters of each method using cross-validation. Figure 5.3 (a) shows the evaluation results: the percentage of truly liked movies in the top N recommendations where $N = 10, 20, 30, 40$, and 50. SVD and RRPP (denoted as *Regularized SVD*) outperformed all baseline methods. As expected, RRPP performed much better than any other method. It was robust and insensitive to its parameters. Second, both

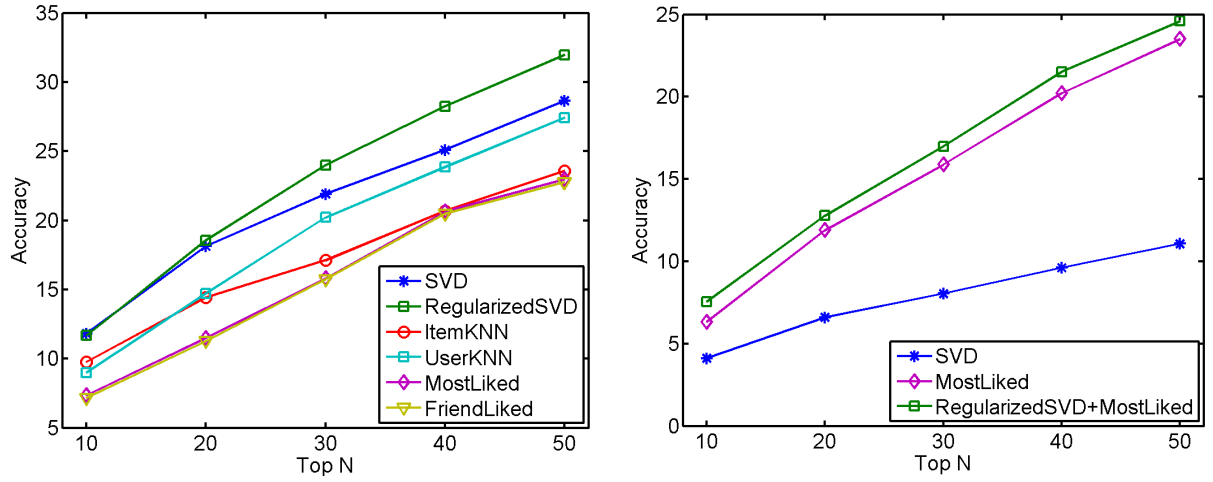


Figure 5.3: Accuracy of top-N movie recommendations. (a) Inductive stream reasoning and deductive stream reasoning separately. (b) Inductive and deductive stream reasoning combined.

Item kNN and *User kNN* curves are also above the baselines, suggesting that users and items share some common regularities. For example, users who like the same actors are likely to watch movies featuring them. Of course, SVD and RRPP exploited such common regularities as well. Third, the two baseline methods *MostLiked* and *FriendLiked* almost completely overlapped, perhaps because most users tend to watch select movies based on their general popularity rather than their friends' preferences.

In the second part of our empirical study, we experimented with combining the output produced by the deductive and the inductive reasoners. In this scenario, the inductive reasoner modeled long-term user preferences and was trained only on data more than 30 days old, reflecting the reasonable assumption that the long-term model should be updated only at larger intervals because required computations can be quite costly if we avoid subsampling. The deductive reasoner contributed predictions in the form of most liked by all users, which aggregates recent recommendations to capture short-term trends. Strictly speaking, the deductive reasoner utilizes an inductive process, but the querying language C-SPARQL inherently supports aggregation. Figure 5.3 (b) clearly shows that the combination of the long-term inductive model and the "most liked" deductive model outperformed each model when considered separately. Note that the short-term trends identified in our experiment could have been predicted by multivariate analysis. In practice, for a given use case, it is flexible in choosing deductive and inductive reasoners. Experiments with more sophisticated hype cycle models and exploring different combination schemes will be investigated

in future work.

5.2 Life Science: Disease-Gene Prioritization

Life science data constitutes a significant part of the Linked Open Data cloud. To a large extent, these data have been extracted from well-maintained databases, and the quality of the data is relatively high as a result. We apply the SUNS framework to an essential problem in life science: disease-gene prioritization.

5.2.1 The Problem

Disease genes are genes that either cause or are associated with particular diseases. More than 2,500 disease genes have been discovered to date, but the relationship between genes and disease is quite complicated since most diseases are polygenic and exhibit different clinical phenotypes. High-throughput genome-wide experimental techniques such as linkage analysis and gene expression profiling typically identify hundreds of candidate genes potentially involved in a given disease, and it remains challenging to identify true disease genes among them: genes often perform multiple functions, and mutational analyses of a particular gene can reveal dozens of mutational sites that associate different phenotype with diseases like cancer [51]. Analyzing the gene-disease relationship becomes even more complicated when considering environmental and physiological factors, as well as exogenous agents like viruses and bacteria.

Due to the complexity of associating genes with diseases, in many applications, including medical diagnosis, prognosis, and personalized treatment of diseases, it is crucial for researchers to be able to rank genes according to their (estimated) relevance to a given disease. In recent years, a number of tools have been developed for this purpose, such as ToppGene [18] and similar approaches that use feature-based measures of similarity between genes to discover unknown disease genes by making generalizations about known ones. Kann [51] reviews the advances in the field of translational bioinformatics, focusing on advances in computational techniques of searching for and classifying disease genes, as machine learning methods such as decision trees and similarity-based methods are mainly used. For example, PhenoPred generates a similarity score that represents the chance that a gene-disease association is real [84]. In the following section, we will compare SUNS with the ToppGene Suite, which is one of the state-of-the-art approaches to disease gene prioritization with an easy-to-use interface [18], and which relies on a fuzzy measure of similarity between genes.

Certain properties of disease genes differentiate them from all others, and those properties have been exploited by computational tools that prioritize disease gene candidates identified by laboratory experiments. All current tools are based on the integration of different properties, including:

- gene function: disease genes are expected to share common functional properties,
- pathways: disease genes likely share common pathways,
- gene expression: disease genes are thought to be co-expressed,
- gene regulation: genes within the same gene-regulation network are expected to affect similar diseases
- sequence properties and
- protein interaction: disease genes are often highly connected with other genes from the same disease.

The quantity and quality of data generated by laboratory experiments is a major limitation of existing disease gene prioritization techniques. For instance, methods of prioritization based on protein-protein interactions suffer from incomplete and low-quality data currently available on protein interaction networks in mammals. Disease mapping information used to train and evaluate computational methods introduces additional uncertainty, because the resolution of that information varies greatly, and its use leads to a large number of false-positive associations between genes and diseases.

5.2.2 Empirical Study

Gene-Disease Data

A great benefit of LOD data is the ease with which the data required for our empirical study, i.e., gene-disease relationships, gene attributes, and disease attributes, can be extracted. In our empirical study, we exploited manually-curated and well-maintained databases of gene-disease relationships. More specifically, we used OMIM [36], UniProt [3], PharmGKB [39] and CTD [73]. All of these databases except UniProt use Entrez Gene [71] identifiers for genes, and the mapping of UniProt to Entrez Gene is complete enough that it was possible to use only Entrez Gene identifiers as the vocabulary of genes. By contrast, data on diseases were much less standardized. Some of the databases selected for our study

use OMIM terms, while others use MeSH. The mapping of OMIM terms to MeSH terms for diseases (or vice versa) is non-trivial and the subject of ongoing research (e.g., [59]). For this reason, we utilized only data with MeSH identifiers in our experiments. After extracting gene-disease relations from the databases, we stored them as triples of the form `(<Entrez Gene ID>, siemens:related_to, <MeSH ID>)` in an RDF triple store.

In the next step, we retrieved data on gene attributes from Bio2RDF [10] and linked life data (LLD)[4] in particular. Bio2RDF and LLD are two projects from the Semantic Web and Linked Data communities that integrate various bioinformatics databases and publish this information in the form of RDF triples. Bio2RDF Release 3 (July 2014) [5] contained roughly 11 billion triples across 35 datasets. Figure 5.4 shows the various databases and how they are connected.

In Bio2RDF, resources are accessible via a REST-like interface and identified by normalized URIs of the form *http://bio2rdf.org/<namespace:id>*. For each Entrez Gene ID, we, therefore, queried Bio2RDF using appropriate identifiers and retrieved all available information as RDF triples. In this way, we were able to derive attributes of protein interactions by using data from BioGRID[94] and HPRD [81], information on gene function using Gene Ontology annotations [10], and information on gene pathways from Pathway Commons, UniProt, [3] and Reactome [49]. Additionally, we included information about proteins from CDD [72] as well as PubMed co-citations. Gene lengths ware added in a postprocessing step as a normalized continuous attribute. As a result, we were able to retrieve most of the attributes identified in [50] as important for gene-disease prioritization by making a single request to Bio2RDF and performing one postprocessing step for any given gene in the gene-disease relation graph.

Finally, we gathered data for disease attributes. As mentioned above, we used only MeSH terms for diseases for gene-disease relationship prediction in our experiments. Disease attributes are neither easily accessible nor readily available in any RDF format. We crawled the MeSH tree structure of diseases via the MeSH Browser[6], then split node numbers according to the levels of the MeSH hierarchy from the top level to the leaves. For instance, the disease "Abetalipoproteinemia" (mesh:D000012) is located in three nodes in the hierarchy, one of which is C16.320.565. From this, we define the attributes C16, C16.320, and C16.320.565. In this way, we obtained in total 4,389 attributes for 1,138 diseases arranged in a 10-level hierarchy.

The gene-disease data set is available at

[4] http://linkedlifedata.com
[5] https://github.com/bio2rdf/bio2rdf-scripts/wiki
[6] https://meshb.nlm.nih.gov/search

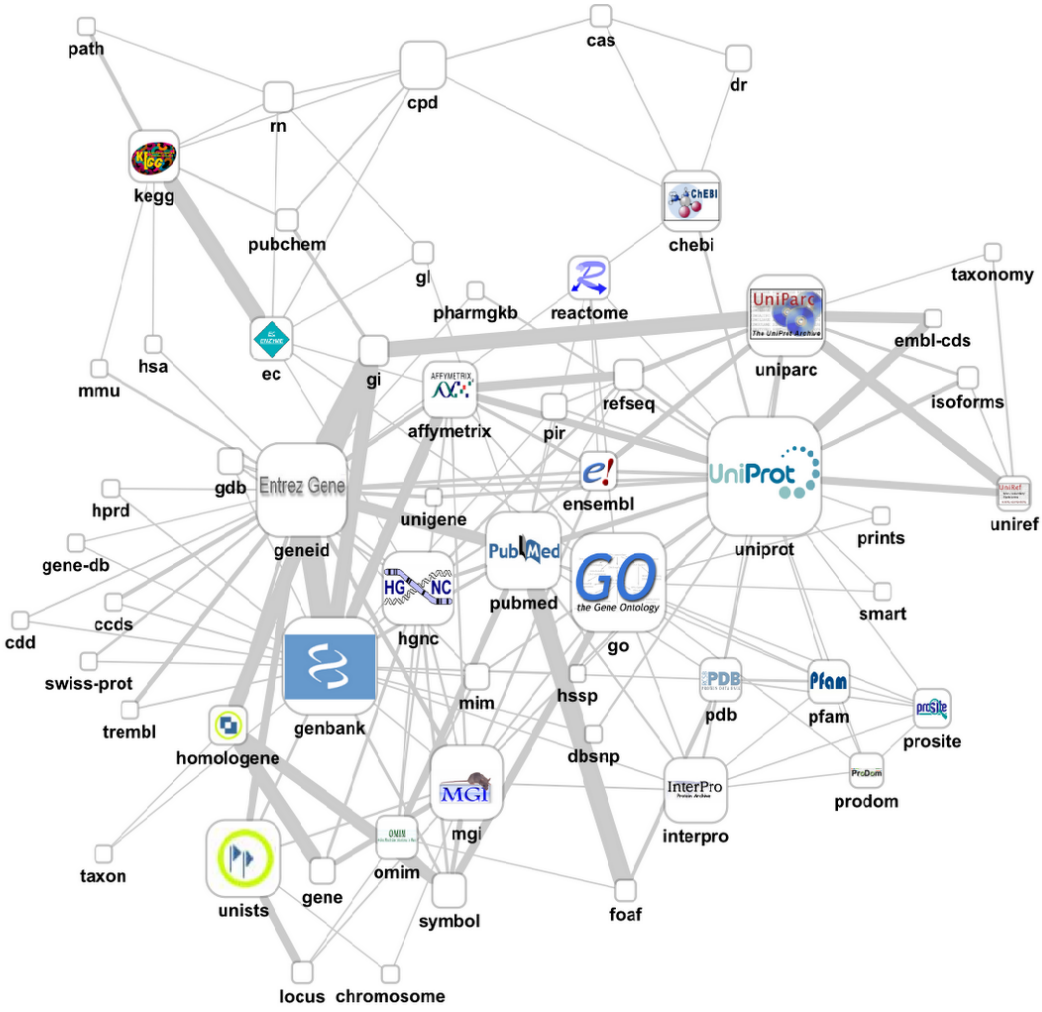


Figure 5.4: Bio2RDF databases and connections.
(Source http://bio2rdf.blogspot.com/2008/)

http://www.dbs.ifi.lmu.de/~huang/index.html#datasets.

The Data Matrices

We compared the results we obtained from in two experiments.

In the first experiment, genes were treated as statistical units. The resulting data matrix consisted of two parts Y and X_G. The submatrix Y was a $N \times M$ matrix where N genes form the rows, and the columns represent the M diseases. In Y, an element $y_{i,j}$ is equaled *one* if it is known that gene i affects disease j and *zero* otherwise. The submatrix X_G contained the attributes of the genes.

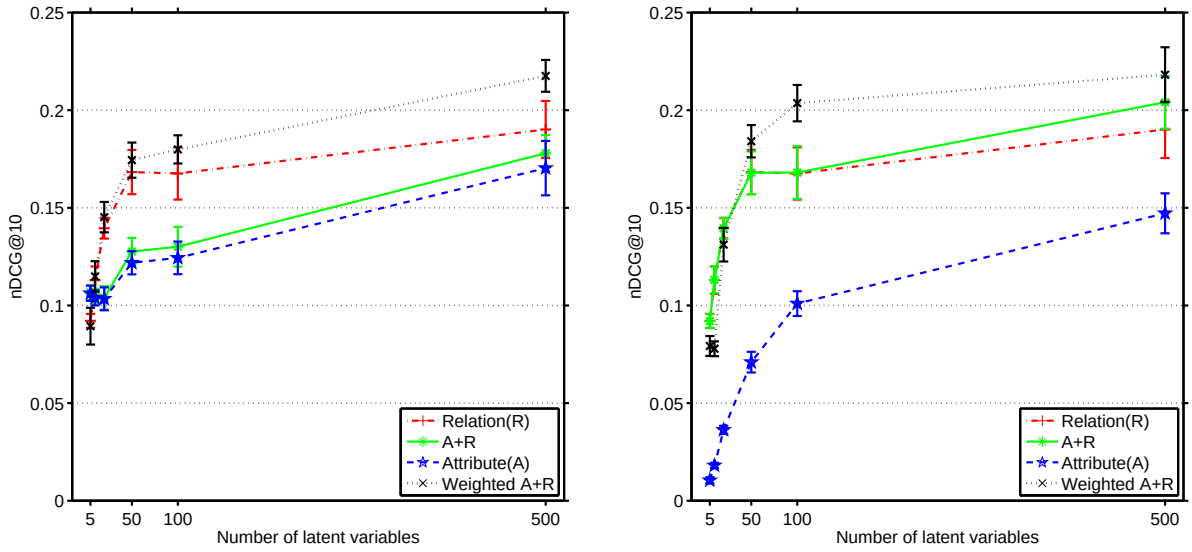
Using that data matrix, we explored 3,820 genes and 3,102 diseases, of which 1,138 corresponded MeSH terms. Y was very sparse and contained only 0.07% *ones*, while the MeSH part of Y had 0.13% *ones*. Originally, we obtained almost a million attributes. This number was reduced to somewhat fewer than 100,000 after culling attributes associated with only one single gene. X_G was also very sparse, containing 0.13% *ones*.

In the second experiment, diseases were treated as statistical units, and the resulting data matrix consisted of two submatrices $Y^\top$ and X_D. X_D contained 4.389 attributes of the diseases. Only 0.28% of the entries in X_D were *ones*.

Evaluation

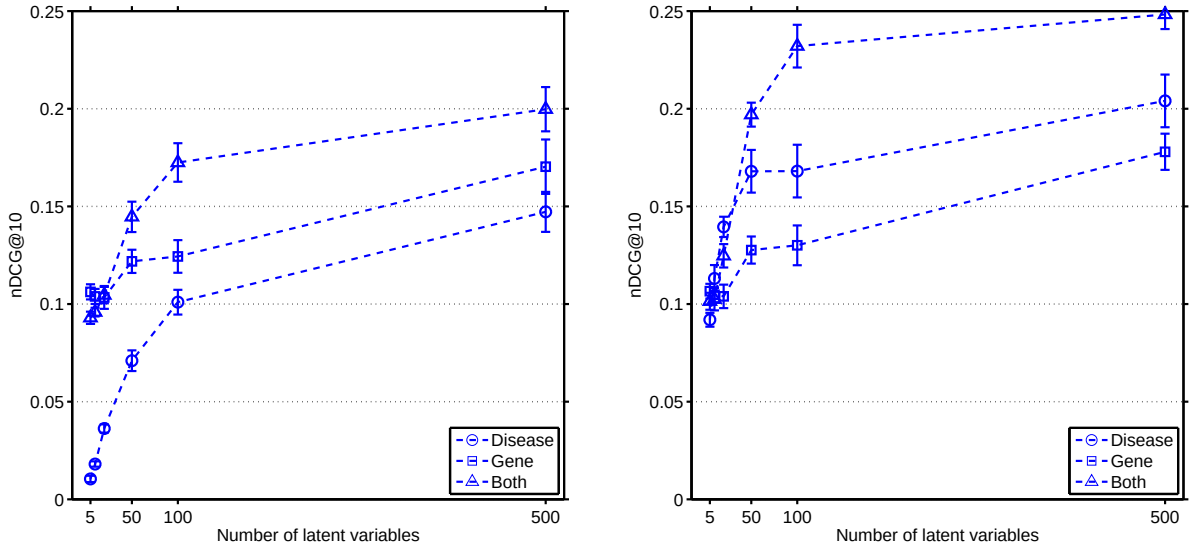
In our experiments, we hid a known gene-disease relation for each gene we evaluated, treated this relation as unknown by setting the entry to *zero*. We then applied the kernelized SUNS approach in order to predict the likelihood of all unknown gene-disease relations being true. Finally, we evaluated the validity of the predictions our approach made for the real gene-disease relations we removed from the data by using an nDCG@n score with $n = 10$ (described in Appendix 6). We repeated the procedure five times in order to produce error bars and mean values.

Figure 5.5 (a) and (b) show the nDCG score against rank r for the SUNS models applied to the gene data set and disease data set, respectively. In both experiments, pure attribute-based predictive models *Attribute(A)* (only using X_G, resp. X_D for the kernel computations, equivalent to regularized PCA regression in the covariate space described in Section 3.2) produced the worst results. Models using Y for the kernel *Relation(R)* (equivalent to regularized matrix reconstruction utilizing kernel PCA) performed much better. When both Y and X_G, resp. X_D were exploited through the *A+R* approach (kernel RRPP) and when using the weighting parameter $\alpha = 1$ (meaning that the two sources-the relations and attributes-were equally considered), the results were suboptimal,



(*a*) gene data set (*b*) disease data set

Figure 5.5: (a) The nDCG@10 score against rank r for the models where genes served as statistical units. The bottom (blue, Attribute(A)) line shows the performance of the model using only X_G for the kernel and the second line from the top (red, Relation(R)) shows the performance of the model using Y for the kernel. We can see that the gene-disease relations served as a better predictor. The RRPP model with $\alpha = 1$ (second line from bottom, green, A+R) performed worse than the relation-based model. The RRPP SUNS model with a tuned $\alpha = 0.1$ gave the best results. (b) The results of the models where diseases served as the statistical units. At high rank r the attributes were quite informative. The RRPP model with $\alpha = 0.5$ gave the best results, when rank r exceeded 50.



(a) attribute only

(b) attribute and relationship

Figure 5.6: (a) The nDCG@10 score against rank r for the multi-population models with equal weights and the models using only attributes X_G, resp. X_G for calculating the kernel. The results demonstrate that the multi-population model gave the best results. (b) The results of the RRPP models. The multi-population model again achieved the best results overall.

since the attributes describing genes resp. diseases were dominant due to their greater number. *Weighted A+R*, which carefully tuned α achieved the best performance. In the first experiment, the best results were obtained using $\alpha = 0.1$, and in the second experiment by using $\alpha = 0.5$. RRPP with a properly tuned α demonstrated similar performance in both experiments. Figure 5.6 (a) shows the nDCG score against rank r for the multi-population SUNS models in which the predictions of both models were simply averaged. In Figure 5.6 (b), RRPP gave the overall best performance by just averaging predictions when it exploited both gene-diseases relations and the attributes of genes and diseases.[7]

Additionally, we compared SUNS with ToppGene in connection with four diseases: Autistic Disorder, Psoriasis, Hypertension, and AIDS. We followed the same evaluation procedure as described in [19] and [1]. For each disease, in addition to related genes, we selected 99 unrelated genes at random. We performed leave-one-out validation: in each run, we treated a truly related gene as unrelated, the target gene, and prioritize it together with those 99 unrelated. On the one hand, we applied SUNS to the data in order to prioritize unrelated genes (including 99 unrelated genes and the target one), and on the other hand, we submitted all training and test data through the ToppGene web interface[8] and recorded the resulting prioritization of those genes. Finally, we evaluated the quality of each method's prioritization according to the frequency at which they ranked target genes above a given threshold, e.g., among the top 5%, 10%, and 20% of the results. The predictions generated by SUNS were superior to ToppGene in most cases, and for a higher threshold, ToppGene tended to be reliable. We believe that these results are quite promising, especially in light of the fact that the weights given to various different attributes were carefully adjusted in ToppGene, whereas SUNS weighted all attributes equally. We believe that our SUNS approach can realize even better performance by carefully tuning the weights of attributes.

5.3 Location-based Personalized Recommendation

The rapid growth in the publication of personal opinions through microblogging venues such as Twitter has supported the creation of novel, emerging social and commercial services in recent years. In augmented reality (AR) applications such as BOTTARI, for instance, the information generated by microblogging makes it possible to leverage the

[7] We did not observe any improvement when varying a weight factor between the models, compared to a simple average.

[8] http://toppgene.cchmc.org/prioritization.jsp

(temporally weighted) opinions of a user's community in order to provide her with personalized, localized recommendations for points of interest (POIs) she might wish to visit. The technological basis for BOTTARI's recommendations is the LarKC platform [62], a highly scalable platform for rapid prototyping and development of Semantic Web applications, and a recommendation engine relying on LarKC's deductive and inductive stream reasoning.

In this section, we describe the use of SUNS. We evaluate the quality of recommendations on POIs based on tweets spanning a three-year period and pertaining to 319 restaurants located in the Insadong district of Seoul, a popular 2 km^2 tourist area within the South Korean capital.

5.3.1 The BOTTARI Mobile Application

BOTTARI[9] is an augmented reality (AR) application on Android that draws the user's attention to points of interest (POIs) in the user's current vicinity, and particularly to restaurants and dining establishments. Rather than simply showing nearby businesses, BOTTARI provides personalized recommendations based on the local context and ratings of the POIs derived by the analysis of microblogs. Recommendations are made using a combination of inductive and deductive stream reasoning techniques, and when a user selects a particular POI, the application reveals detailed information about it, including its reputation. A video demonstration of BOTTARI's mobile application running on a mobile phone and a tablet is available on YouTube [10].

The application offers four types of recommendations:

- *for_me* gives personalized recommendations suggested by large-scale mobile search studies like [91, 28];
- *popular* bases its recommendations on POI ratings from the last 3 years extracted from microblogs;
- *emerging* focuses on ratings from the last month; and
- *interesting* returns POIs based on a category of interest to the user, e.g., sites or establishments "for tourists".

[9] In the Korean language, "bottari" is a cloth bundle that carries a person's belongings while traveling.
[10] A video of BOTTARI is available on YouTube at http://www.youtube.com/watch?v=c1FmZUz5BOo.

For the purposes of this thesis, we will focus here exclusively on BOTTARI's recommendation engine, and more particularly, on the application of SUNS to *for_me* recommendations. Other components of the application, such as its stream crawler, opinion miner, and geospatial knowledge base, lie outside the scope of our work.

5.3.2 Data and Ontology

As mentioned above, BOTTARI's recommendations to its users are based on the analysis of microblogs published in Korea. In this section, we describe the data and ontology used in the application.

The Insadong area is a two-square-kilometer district in Seoul with a high density of restaurants. BOTTARI's data on 319 of the restaurants in this area were collected from Yelp[11], PoiFriend[12], Yahoo! Local[13], TrueLocal[14], Korean restaurant Web sites and several Korean portals. The resulting high-quality, geo-referenced knowledge base describes each restaurant using 44 attributes, such as name, images, position, address, ambiance, specialties, and categories.

The restaurants' reputations and customer ratings were gathered from 200 million related tweets via Twitter's API over the course of 3 years, from February 4, 2008, to November 23, 2010 (1,023 days). After applying the opinion mining technique, tweets not expressing an opinion on any POI were discarded, and 110,000 tweets produced by more than 31,000 users were kept (see Table 5.1 for statistics of the collected data set).

The temporal distribution of tweets is shown in Figure 5.7(c). Most tweets (85%) were collected in the final six months of the three-year period. Concrete numbers of tweets in different time frames are shown in Table 5.2. The exponential growth in the number of tweets over time is due to different reasons: *a*) the tweets from 2008 were collected one year later, in 2009, and it was difficult to gather those messages because of the "oblivion" in Twitter stores; *b*) the usage of Twitter became mainstream in Korea only in 2009; and finally, *c*) the crawling algorithm was changed and improved in April 2010.

In addition to the number of entities, i.e., users and POIs, Table 5.1 also shows the numbers of positive, negative, and neutral ratings applied to all POIs. Based on these statistics, we can see that the data set possesses the following characteristics:

- *High sparsity*: Only 1.42% of POIs were rated per user. When sparsity is defined as

[11]Cf. `http://www.yelp.com/`.
[12]Cf. `http://www.poifriend.com/`.
[13]Cf. `http://local.yahoo.com/`.
[14]Cf. `http://www.truelocal.com.au/`.

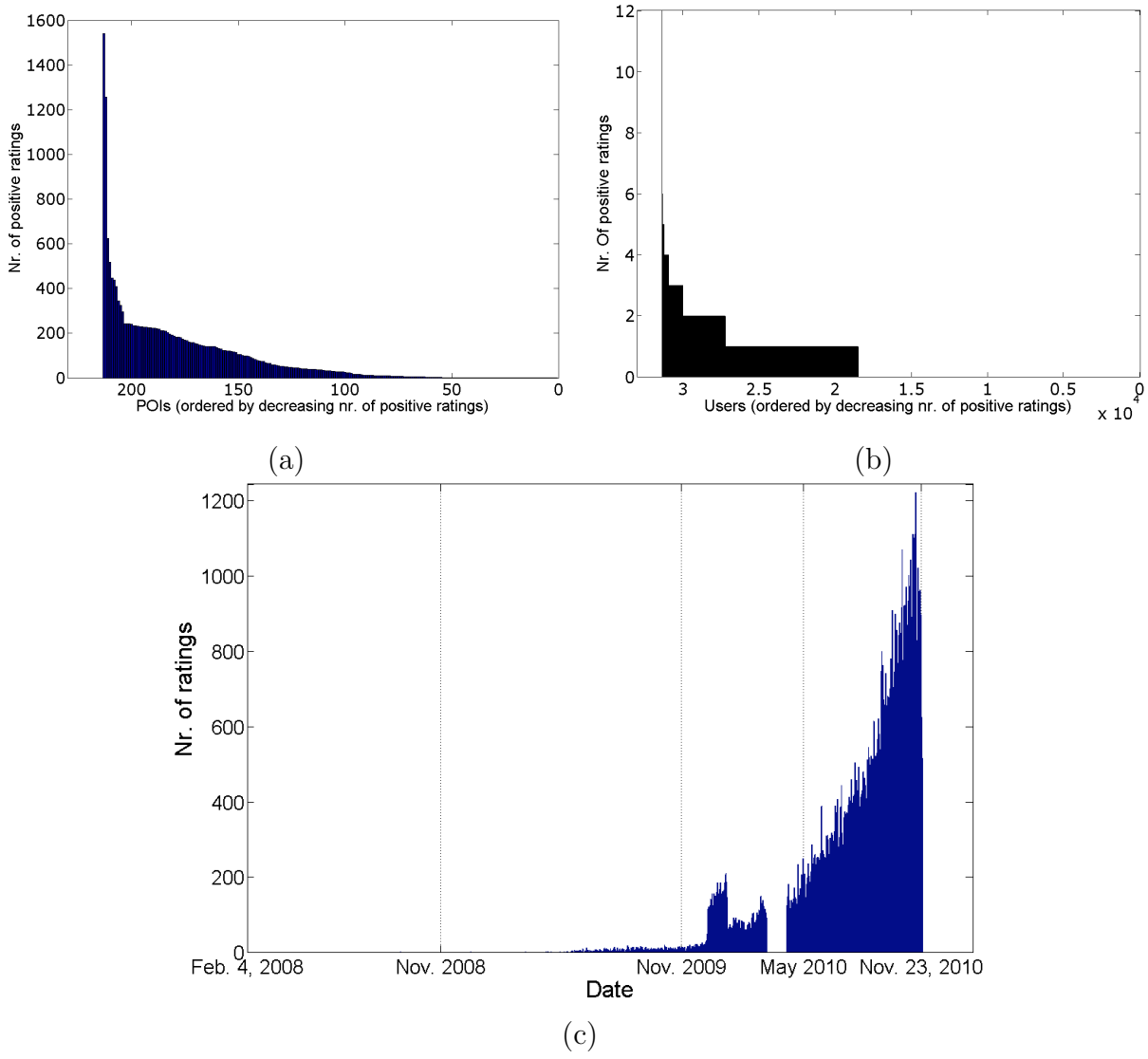


Figure 5.7: Dataset statistics: (a) positive ratings by POIs, (b) positive ratings by users and (c) distribution of tweets over time.

	Type		#POI	#User	Sparsity
#Ratings	Positive	19,045	213	12,863	99.30%
	Negative	14,404	181	10,448	99.24%
	Neutral	75,941	245	28,056	98.90%
	Total	109,390	245	31,369	98.58%

Table 5.1: Statistics of the data set showing the number of positive, negative, and neutral ratings respectively and the number of users and restaurants involved rated in each way. For instance, 10,448 users gave 14,404 negative ratings of181 POIs, and the data matrix representing these negative ratings therefore is a 10,448-by-181 matrix where 14,404 entries are non-zero, and the remaining 99.24% entries are zeros (sparsity).

	Nr. of ratings	%
Last day	188	0.17
Last 2 days	703	0.64
Last 7 days	5,057	4.62
Last 30 days	27,049	24.73
Last 90 days	65,600	70.01
Last 180 days	93,696	85.65
Total	109,389	100.00

Table 5.2: Number of ratings in different time frames

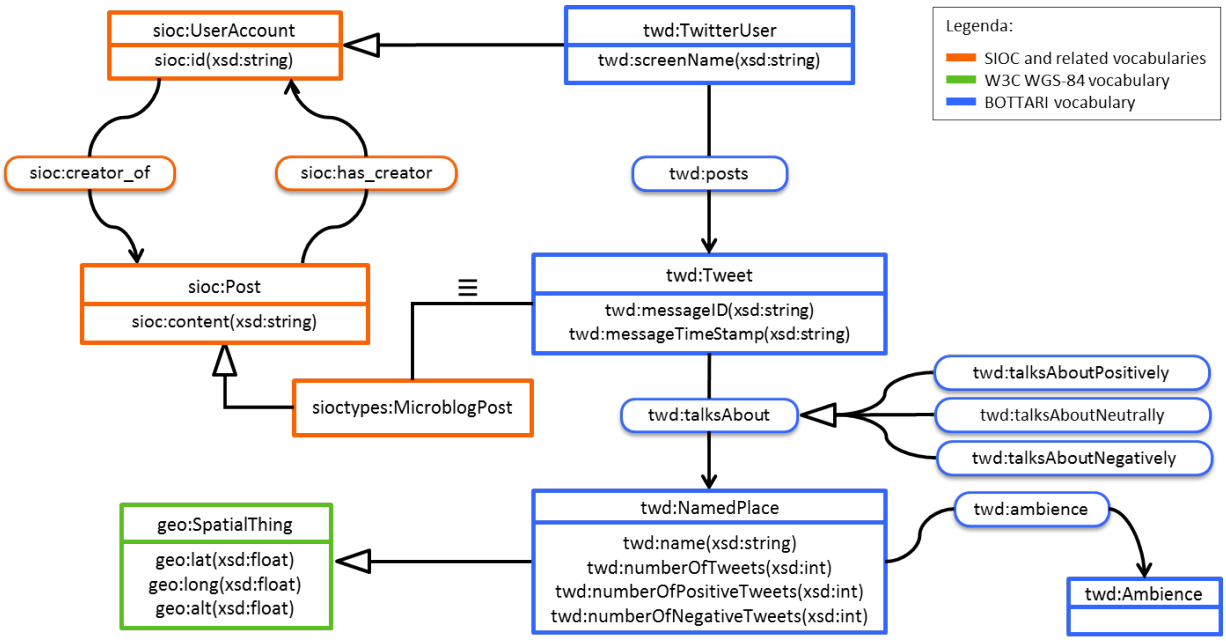


Figure 5.8: Ontology modelling of the data.

$sparsity = 1 - \frac{\#Ratings}{\#POIs \times \#Users}$, the sparsity of positive ratings is higher than 99.3%,

- *Incompleteness*: While some POIs have neither positive nor negative ratings, a great number of users provided neither positive nor negative ratings. For example, only 41% of users rated at least one POI positively.

- *Multiple ratings*: A user can rate a particular POI several times expressing different opinions each time. In our experiments, positive, negative, and neutral ratings are represented as separate statements $positive(u, p)$, $negative(u, p)$ and $neutral(u, p)$. The statement $positive(u, p) = 1$ means that at least one positive rating was given by user u on POI p. If there is no positive rating of user u for POI p, $positive(u, p) = 0$. The same approach is applied to $negative(u, p)$ and $neutral(u, p)$.

In addition to the temporal distribution shown in Figure 5.7(c), we gathered statistics on ratings across users and POIs in order to better understand the data distribution. The distribution of positive ratings over POIs and users, plotted in Figure 5.7(a) and (b), respectively, is of particular interest. On average, each user gave 1.5 positive ratings, and 89 users rated POIs positively. 31 POIs received more than 200 positive ratings. Those POIs can be considered the most liked or most popular.

The attributes of POIs and microblogs were first converted into RDF triples and RDF streams, respectively. These RDF-ized data were modeled according to the ontology represented in Figure 5.8. The BOTTARI Ontology is an extension of Semantically-Interlinked Online Communities (SIOC) [8] – a commonly used vocabulary for expressing the user-generated content of an online community. BOTTARI's ontology takes from SIOC the relation between Twitter users and their tweets and enriches it by further describing the relation between a tweet and its topic (i.e., the POI cited in the tweet[15]). The ontology adds the opinion expressed by the user about the topic through the *twd:talksAbout* property – and its sub-properties for positive, negative, and neutral opinions.

An example of an RDF stream in BOTTARI is given in List 5.5. Each RDF triple uses the ontology described above and is annotated with the tweet's timestamp. The notation used here is defined in [9].

```
1 (<:userID twd:posts :tweetID >,                2011-10-12T13:34:41)
2 (<:tweetID twd:talksAboutPositively :poiID >, 2011-10-12T13:34:41)
```

Listing 5.5: An example of an RDF stream.

The geographical information of the ontology is introduced by modeling the POIs as *spatial things* (according to the WGS84 vocabulary[16]) and enriching them with information in additional categories, such as an "ambiance" category describing the atmosphere of the restaurant and the dynamic count of positive, negative, and neutral ratings of POIs.

5.3.3 Empirical Study

Using the data set described in Section 5.3.2, we tested the personalized recommendation engine *for_me* which was based on SUNS . The goal of the study was to evaluate the quality of recommendations made by SUNS in comparison with the quality of recommendations made by some baseline methods.

Methodology

Our evaluation involved first establishing three baselines:

- *random guess (Random)*: recommends a list of restaurants sorted by random order.

[15]Only very few tweets talk about more than one POIs.
[16]Cf. http://www.w3.org/2003/01/geo/.

- *k-nearest neighbor (KNNItem)*: recommends a list of similar restaurants. Let P be the number of POIs and U be the number of users. For the baseline KNNItem, we used the cosine similarity measure on POIs defined as $similarity(p_i, p_j) = \frac{\langle p_i, p_j \rangle}{\|p_i\| * \|p_j\|}$, where p_i and p_j represent the vectors of ratings of the i-th and the j-th POIs given by all users for $i, j \in \{1, \ldots, P\}$, and where $\langle \cdot, \cdot \rangle$ is the scalar product of two vectors and $\| \cdot \|$ is the 2-norm of a vector. We set k to the total number of the POIs.

- *most liked items (MostLiked)*: by means of a predefined SPARQL query, takes all positive ratings in the total three-year period into account to recommend the most positively rated POIs to every user.

First, we compared the performance of Suns with the baseline methods. We then examined the performance of the combination of both inductive and deductive stream reasoning, i.e., Suns and MostLiked using two evaluation methods:

- Normalized discounted cumulative gain (nDCG) (described in Appendix 6) In our experiments, we evaluated ranked lists of POIs and reported nDCG@all scores. In each run, there was only one POI for testing for each user. An NDCG score of *one* means that the test POI is ranked first in the list of POIs, while a low nDCG value near to *zero* indicates that the test POI falls at the bottom of the list.

- Accuracy at the top N POIs $Acc@N = |TP_N|$, where TP_N are true positives in the top N recommendations. In our case, since we have just one test data point per user, Acc@N is equal to either *one* or *zero*.

For both evaluation measures, we averaged the scores across all ranking lists (one list per user).

Settings

We performed our evaluation in two different settings:

- Setting 1: We randomly removed one positive rating for each user and treated it as a test data point. We trained the models using the remaining ratings, then estimated the ratings of the test data points. This setting corresponds to the common method of splitting the data into training and test subsets. We repeated this data split five times.

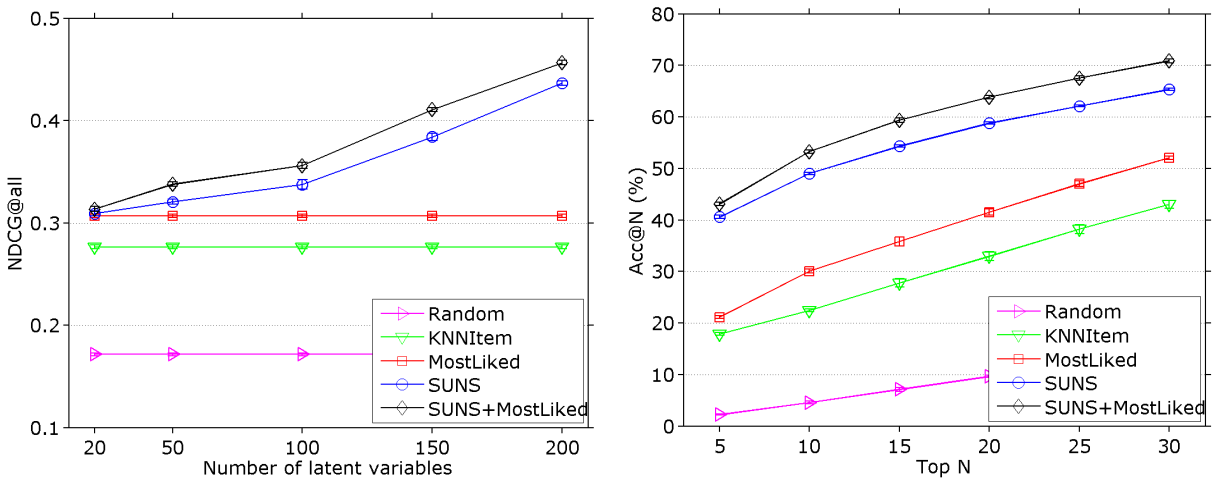


Figure 5.9: Evaluation in Setting 1: nDCG scores (left) and accuracy values at top N (right).

- Setting 2: We hid the newest rating for each user as test data. As a new baseline, TimeWindow was introduced, which corresponds to MostLiked applied to different time frames: 1 day, 2 days, 7 days, 30 days, 90 days, and 180 days[17]. For each time frame, we removed the corresponding older ratings. Table 5.2 shows the number of ratings that remained for training.

We implemented the baselines, Suns and the evaluation methods in Matlab using a laptop with a 2.1GHz CPU and 3.24 GB of RAM running Windows XP, which corresponds to a 50€/month share in a cloud environment[18].

Results

Figure 5.9 shows the results obtained in Setting 1. On the left, the nDCG scores of the tested methods are plotted against the number of latent variables. Since the baselines are independent of this number, they produced three horizontal lines each for Random, MostLiked, and KNNItem. We evaluated Suns with 20, 50, 100, 150 and 200 latent variables. As expected, Random performed worst. MostLiked proved slightly more effective than KNNItem. Its superior performance may be due to the "bandwagon effect"[19] that exists in many social communities. Suns significantly outperformed all baselines when

[17]This baseline was evaluated using the network of C-SPARQL queries in continuous execution.

[18]The calculation of the cost per month was done using https://www.gandi.net/hosting/vps

[19]Bandwagon effect is a form of groupthink known in behavioral science and also a common phenomenon in the daily life: People follow majorities without caring for evidence and their individual preferences.

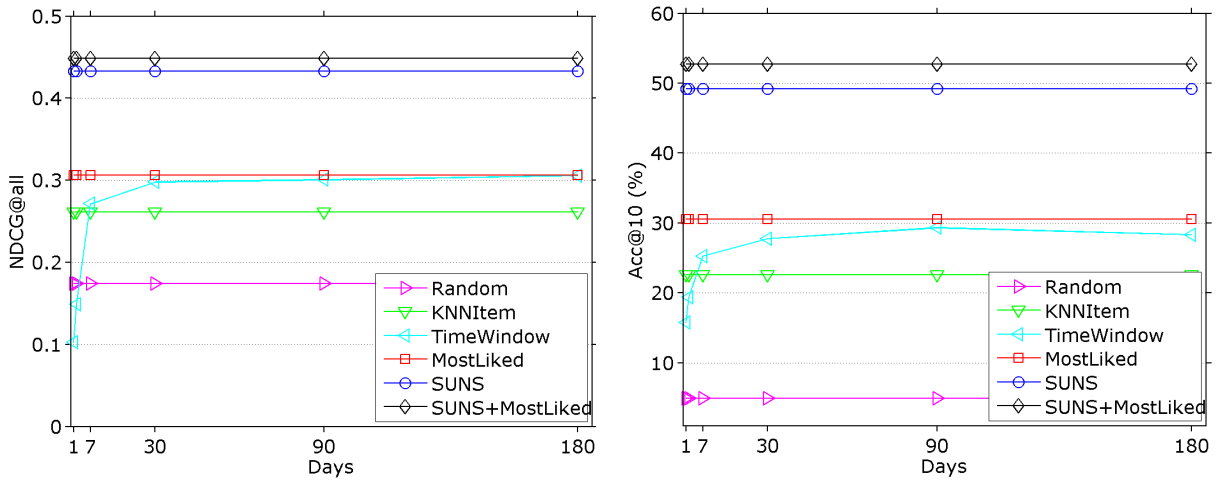


Figure 5.10: Evaluation in Setting 2: nDCG scores (left) and accuracy values at top 10 (right).

the number of latent variables exceeded 100, and the best ranking overall was produced by the combination of both SUNS and MOSTLIKED. These results confirm the idea that a combined approach of deductive and inductive stream reasoning is more effective than other methods. Figure 5.9 shows the accuracy of the top N POIs at right, where $N = \{5, 10, 15, 20, 25, 30\}$. Note that the error bars are so small that they lie within the shapes representing the methods.

In Setting 2, as mentioned previously, we varied the size of the time window from 1 to 180 days. Figure 5.10 plots the nDCG scores on the left and the accuracy at the top 10 on the right. Naturally, only TIMEWINDOW is sensitive to the time window, and it approaches the performance of MOSTLIKED at and above a time window of 90 days. This suggests that using data from a 90-day window is nearly as effective as using the full historical data.

We additionally evaluated the recommendations given by Rough Guide and Trip Advisor. Table 5.3 shows that these two experts and the combination of them performed as poorly as the random guess baseline and were not comparable with SUNS at all.

In our evaluations, SUNS demonstrated excellent scalability. The training process took approximately 86 seconds with 200 latent variables, and the recommendation of POIs cost on average less than five milliseconds of computational time per user (see discussion on scalability in Section 3.3.1). In addition, we observed that SUNS was robust and insensitive to the number of latent variables. Its robustness can be explained by the fact that SUNS is regularized, in contrast to other matrix factorization methods such as Singular Value Decomposition (SVD). Regularization can simplify the use of SUNS, particularly for users

	Trip	Rough	Comb	Random	SUNS
Setting 1					
NDCG@5	0.0087	0.0149	0.0197	0.0134	0,3300
Acc@5 (%)	0.99	1.81	2.53	2.55	40.58
Setting 2					
NDCG@5	0.0072	0.0152	0.0190	0.0136	0.3265
Acc@5 (%)	0.77	1.81	2.55	2.55	40.67

Table 5.3: Comparisons of Trip Advisor (Trip), Rough Guide (Rough) and their combination (Comb) with random guess and SUNS according to nDCG score and accuracy.

with limited expertise in machine learning.

Chapter 6

Conclusions and Future Work

The aim of this thesis was to study the prediction of unknown but potentially true relations by using statistical relational learning and exploiting the regularities of the structured data.

We began by describing the Semantic Web and its standard RDF data model. As we noted, the increasing volume of data in knowledge bases such as the Linked Open Data initiative and the Knowledge Graph offers exciting possibilities and challenges for machine learning algorithms, including the vitally important issue of scalability.

We then discussed ontology-based transductive reasoning, which can be used to infer implicit statements, and reviewed related work in various areas, including statistical learning, Inductive Logic Programming, Relational Graphical Models, and matrix factorization-based learning approaches. In the area of statistical learning, a global model can efficiently deal with missing information but might not be scalable, while conditional models scale better but encounter difficulties when faced with missing information. As a result of our investigation, we set out to develop a model combining global models' ability to handle missing data and conditional models' scalability.

To that end, we introduced a well-defined statistical framework, which we call *Statistic Unit Node Set (SUNS),* capable of systematically generating data matrices, including random variables and fixed covariates based on statistical units' neighborhoods. We then proposed *Reduced Rank Penalized Regression (RRPP)*, a novel multivariate algorithm for relation prediction. RRPP has two advantages over other algorithms: first, it can efficiently learn a global model exploiting all features of data and handle with missing values naturally; second, the algorithm combines regularization and low-rank matrix factorization, which makes it generalizable to statistical units inside and outside of a given sample. In our experiments, this new algorithm was evaluated using a social network data set to predict friendship between persons in both a transductive setting and an inductive set-

ting. Compared to other benchmark methods, RRPP demonstrated promising predictive performance and a high capability of generalization.

We then extended SUNS to become a general kernel approach using the Nyström approximation and demonstrated the scalability of the overall approach, which is guaranteed by the ability to control the number of instances considered in the kernel computations as well as the rank of matrix decomposition. Moreover, as we noted, one can control the number of local features used to derive the kernel. This kernelized approach was then applied to DBpedia, a vast and rich semantic knowledge base, presently consisting of at least 13 billion pieces of information (RDF triples).

Based on SUNS, we established the R-model, a probabilistic model in which statistical units are defined by object-to-object relationships. By applying this model to a movie recommendation system in a social network, this thesis explained how contextual information could be leveraged in a probabilistic way in order to improve the quality of recommendations. The example of movie recommendations illustrated a significant advantage of our model: namely, its modularity, which allows modeling of domains with complex relationships and many variables. The probabilistic model utilized regularized matrix factorization to approximate the local probability distributions. In future work, our approach could be extended to model local interactions between more than two many-state variables by employing the tensor factorization technique [56] for local models. Our experimental results also illustrated R-model's robustness and insensitivity to parameters. These properties can be explained by the fact that in contrast to other matrix factorization approaches such as SVD, the R-model is regularized. As a result, the R-model is relatively easier to use, particularly by those with limited machine learning expertise.

Finally, this thesis described the application of the SUNS approach to three real use cases: semantic social media analysis, disease gene prioritization in the life sciences, and a location-based personalized recommendation system in a mobile app. In the first use case, we introduced an advanced reasoning approach combining deductive and inductive stream reasoning where SUNS served as an inductive stream reasoning method. The inductive reasoner modeled long-term user preferences, while the deductive reasoner made predictions in the form of "most liked" by capturing short-term trends. The experimental results of our investigation of semantic social media analysis showed that the combination of an inductive model and a deductive model outperformed both considered separately. In our analysis of the disease gene prioritization use case, we gathered three data sets: one representing gene-disease relations, one describing attributes of genes, and one containing attributes of diseases from the LOD cloud. We applied the R-model to model those data sets jointly and compared our results with those obtained using ToppGene. The R-model

achieved promising results and was superior to ToppGene in most cases. In the last use case, we integrated SUNS into an AR application providing personalized and location-based recommendations of points of interest (POIs) based on the opinions of social communities. The data sets, in this case, consisted of information collected from various social media networks, including microblogs expressing opinions about restaurants located in a popular tourist area in Seoul. In our empirical study, SUNS demonstrated high accuracy in recommending POIs and excellent scalability. As a point of comparison, we additionally investigated recommendations given by Rough Guide and Trip Advisor and discovered that those recommendations were of bad quality and not comparable with SUNS.

In the future, we believe that the models presented in this thesis can be extended in many ways. As part of our ongoing work in that respect, we will explore additional ways of using SUNS to exploit ontological background information to improve the quality of the results generated by SUNS through various means, such as structuring the learning matrix. Finally, we believe furthermore that our models will be valuable to many other applications in diverse domains and that they open promising new avenues of research on relational learning with large knowledge bases.

Appendices

NDCG

Normalized discounted cumulative gain (nDCG) [48] calculated by summing over all the gains in the rank list R with a log discount factor as

$$NDCG(R) = Z \sum_{k} \frac{2^{r(k)} - 1}{\log(1+k)},$$

where $r(k)$ denotes the target label for the k-th ranked item in R, and r is chosen such that a perfect ranking obtains value *one*. To focus more on the top-ranked items, one often considers *nDCG@n*, which only counts the top n items in the rank list.

NDCG is a standard measure in information retrieval and is used to evaluate the quality of the returned ranked results for a given query. It especially emphasizes the very top positions in R due to the log function on the position k, which means that the nDCG score increases when the position of a test item is lifted from the second position to the first one, much more than when its position is improved from 100 to 99.

Bibliography

[1] Stein Aerts, Diether Lambrechts, Sunit Maity, Peter Van Loo, Bert Coessens, Frederik De Smet, Leon-Charles Tranchevent, Bart De Moor, Peter Marynen, Bassem Hassan, Peter Carmeliet, and Yves Moreau. Gene prioritization through genomic data fusion. *Nature Biotechnology*, 24(5):544, 537, May 2006.

[2] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. The MIT Press, 2004.

[3] R. Apweiler, A. Bairoch, C. H Wu, W. C Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, et al. UniProt: the universal protein knowledgebase. *Nucleic acids research*, 32(Database Issue):D115, 2004.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. *The Semantic Web*, 2008.

[5] Marco Balduini, Irene Celino, Daniele Dell'Aglio, Emanuele Della Valle, Yi Huang, Tony Kyung-il Lee, Seon-Ho Kim, and Volker Tresp. BOTTARI: an augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. *Journal of Web Semantics*, 16:33–41, 2012.

[6] Marco Balduini, Irene Celino, Daniele Dell'Aglio, Emanuele Della Valle, Yi Huang, Tony Kyung-il Lee, Seon-Ho Kim, and Volker Tresp. Reality mining on micropost streams - deductive and inductive reasoning for personalized and location-based recommendations. *Semantic Web Journal*, 5(5):341–356, 2014.

[7] Davide Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Yi Huang, Volker Tresp, Achim Rettinger, and Hendrik Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intelligent Systems*, 99, 2010.

[8] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, and Michael Grossniklaus. An execution environment for C-SPARQL queries. In *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings*, pages 441–452, 2010.

[9] Davide Francesco Barbieri and Emanuele Della Valle. A proposal for publishing data streams as linked data - A position paper. In *Proceedings of the WWW2010 Workshop on Linked Data on the Web, LDOW 2010, Raleigh, USA, April 27, 2010*, 2010.

[10] F. Belleau, M. A Nolin, N. Tourigny, P. Rigault, and J. Morissette. Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5), 2008.

[11] Bettina Berendt, Andreas Hotho, and Gerd Stumme. Towards Semantic Web mining. In *ISWC '02: Proceedings of the First International Semantic Web Conference*. Springer-Verlag, 2002.

[12] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[13] Hendrik Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2), 1998.

[14] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2787–2795, 2013.

[15] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Uncertainty in Artificial Intelligence*, 1998.

[16] Dan Brickley and Libby Miller. *The Friend of a Friend (FOAF) project*. http://www.foaf-project.org/, 2014.

[17] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Computing Research Repository - CORR*, 2008.

[18] Jing Chen, Eric E. Bardes, Bruce J. Aronow, and Anil G. Jegga. ToppGene suite for gene list enrichment analysis and candidate gene prioritization. *Nucleic Acids Research*, 37, 2009.

[19] Jing Chen, Huan Xu, Bruce Aronow, and Anil Jegga. Improved human disease candidate gene prioritization using mouse phenotype. *BMC Bioinformatics*, 8(1):392, 2007.

[20] William W. Cohen and Haym Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*, 1994.

[21] Claudia d'Amato. *Similarity-based Learning Methods for the Semantic Web*. PhD thesis, University of Bari, 2007.

[22] Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Non-parametric statistical learning methods for inductive classifiers in semantic knowledge bases. In *Proceedings of the IEEE International Conference on Semantic Computing - ICSC 2008*, 2008.

[23] L. De Raedt. Attribute-value learning versus inductive logic programming: The missing links (extended abstract). In *ILP '98: Proceedings of the 8th International Workshop on Inductive Logic Programming*. Springer-Verlag, 1998.

[24] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26, 1997.

[25] L. De Raedt and Wim. Van Laer. Inductive constraint logic. In *ALT '95: Proceedings of the 6th International Conference on Algorithmic Learning Theory*, 1995.

[26] Emanuele Della Valle, Stefano Ceri, Frank van Harmelen, and Dieter Fensel. It's a streaming world! reasoning upon rapidly changing information. *IEEE Intelligent Systems*, 24(6):83–89, 2009.

[27] P. Domingos and M. Richardson. Markov logic: A unifying framework for statistical relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[28] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW*, pages 581–590, 2007.

[29] S. Džeroski. Inductive logic programming in a nutshell. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[30] P. Edwards, G. Grimnes, and A. Preece. An empirical investigation of learning from the Semantic Web. In *ECML/PKDD, Semantic Web Mining Workshop*, 2002.

[31] Werner Emde and Dietrich Wettschereck. Relational instance based learning. In Lorenza Saitta, editor, *Machine Learning - Proceedings 13th International Conference on Machine Learning*, 1996.

[32] T. Gärtner, J.W. Lloyd, and P.A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3), 2004.

[33] L. Getoor, N. Friedman, D. Koller, A. Pferrer, and B. Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[34] Stephan Grimm and Boris Motik. Closed world reasoning in the semantic web through epistemic operators. In *OWLED*, 2005.

[35] Marko Grobelnik and Dunja Mladenic. Automated knowledge discovery in advanced knowledge management. *Library Hi Tech News incorporating Online and CD Notes*, 9(5), 2005.

[36] A. Hamosh, A. F Scott, J. S Amberger, C. A Bocchini, and V. A McKusick. Online mendelian inheritance in man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, 33(Database Issue):D514, 2005.

[37] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Myers Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 2000.

[38] Ivan Herman. *Tutorial on the Semantic Web*. W3C, http://www.w3.org/People/Ivan/CorePresentations/SWTutorial/Slides.pdf, 2002.

[39] M. Hewett, D. E Oliver, D. L Rubin, K. L Easton, J. M Stuart, R. B Altman, and T. E Klein. PharmGKB: the pharmacogenetics knowledge base. *Nucleic Acids Research*, 30(1):163, 2002.

[40] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. *Semantic Web*. Springer, 2008.

[41] Yi Huang, Markus Bundschus, Volker Tresp, Achim Rettinger, and Hans-Peter Kriegel. Multivariate prediction for learning on the Semantic Web. In *Proceedings of the 20th International Conference on Inductive Logic Programming (ILP)*, 2010.

[42] Yi Huang, Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A scalable kernel approach to learning in semantic graphs with applications to linked data. In *1st Workshop on Mining the Future Internet*, 2010.

[43] Yi Huang, Volker Tresp, Markus Bundschus, and Achim Rettinger. Scalable relational learning for sparse and incomplete domains. In *International Workshop on Statistical Relational Learning (SRL 2009)*, 2009.

[44] Yi Huang, Volker Tresp, Markus Bundschus, Achim Rettinger, and Hans-Peter Kriegel. Multivariate structured prediction for learning on the semantic web. In *Proceedings of the 20th International Conference on Inductive Logic Programming (ILP)*, 2010.

[45] Yi Huang, Volker Tresp, Maximilian Nickel, Achim Rettinger, and Hans-Peter Kriegel. A scalable approach for statistical learning in semantic graphs. *Semantic Web Journal*, 5(1):5–22, 2014.

[46] Yi Huang, Volker Tresp, and Hans peter Kriegel. Multivariate prediction for learning in relational graphs. In *NIPS 2009 Workshop: Analyzing Networks and Learning With Graphs*, 2009.

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

[48] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[49] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. De Bono, B. Jassal, G. R. Gopinath, G. R. Wu, L. Matthews, et al. Reactome: a knowledgebase of biological pathways. *Nucleic acids research*, 33(Database Issue):D428, 2005.

[50] M. G Kann. Advances in translational bioinformatics: computational approaches for the hunting of disease genes. *Briefings in Bioinformatics*, 11(1):96, 2010.

[51] Maricel G. Kann. Advances in translational bioinformatics: computational approaches for the hunting of disease genes. *Briefing in Bioinformatics*, 11, 2010.

[52] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[53] Kristian Kersting and L. De Raedt. Bayesian logic programs. Technical report, Albert-Ludwigs University at Freiburg, 2001.

[54] Christoph Kiefer, Abraham Bernstein, and André Locher. Adding data mining support to SPARQL via statistical relational learning methods. In *Extended Semantic Web Conference 2008*. Springer-Verlag, 2008.

[55] Atanas Kiryakov. Measurable targets for scalable reasoning. *Ontotext Technology White Paper*, 2007.

[56] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 2009.

[57] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1998.

[58] S. Kramer, N. Lavrac, and P. Flach. From propositional to relational data mining. In S. Džeroski and L. Lavrac, editors, *Relational Data Mining*. Springer-Verlag, 2001.

[59] Isaac Kunz, Ming-Chin Lin, and Lewis Frey. Metadata mapping and reuse in cabig. *BMC Bioinformatics*, 10 Suppl 2:S4, 2009.

[60] N. Landwehr, A. Passerini, L. De Raedt, and Paolo Frasconi. kFOIL: Learning simple relational kernels. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[61] Niels Landwehr, Kristian Kersting, and L. De Raedt. nFOIL: Integrating naïve bayes and FOIL. In M. Veloso and S. Kambhampati, editors, *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005.

[62] LarKC. *The large Knowledge Collider*. EU FP 7 Large-Scale Integrating Project, http://www.larkc.eu/, 2008.

[63] Steffen L. Lauritzen. *Graphical Models*. Oxford Statistical Science Series, 1996.

[64] Nada Lavrač, Sašo Džeroski, and Marko Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *EWSL-91: Proceedings of the European Working Session on Machine Learning*, 1991.

[65] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 1999.

[66] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[67] Francesca A. Lisi. Principles of inductive reasoning on the Semantic Web: A framework for learning in AL-Log. In F. Fages and S. Soliman, editors, *Principles and Practice of Semantic Web Reasoning, Series: Lecture Notes in Computer Science*, 2005.

[68] Francesca A. Lisi. The challenges of the Semantic Web to machine learning and data mining. In *Tutorial at ECML 2006*, 2006.

[69] Francesca A. Lisi. A methodology for building Semantic Web mining systems. In *The 16th International Symposium on Methodologies for Intelligent Systems*, 2006.

[70] Francesca A. Lisi. Practice of inductive reasoning on the Semantic Web: A system for Semantic Web mining. In *Principles and Practice of Semantic Web Reasoning, 4th International Workshop, PPSWR 2006*, 2006.

[71] D. Maglott, J. Ostell, K. D Pruitt, and T. Tatusova. Entrez gene: gene-centered information at NCBI. *Nucleic acids research*, 2006.

[72] A. Marchler-Bauer, J. B Anderson, P. F Cherukuri, C. DeWeese-Scott, L. Y Geer, M. Gwadz, S. He, D. I Hurwitz, J. D Jackson, Z. Ke, et al. CDD: a conserved domain database for protein classification. *Nucleic acids research*, 33(Database Issue):D192, 2005.

[73] CJ Mattingly, MC Rosenstein, GT Colby, JN Forrest Jr, and JL Boyer. The comparative toxicogenomics database (CTD): a resource for comparative toxicological studies. *Journal of Experimental Zoology. Part A, Comparative Experimental Biology*, 305(9):689, 2006.

[74] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4), 1995.

[75] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*. Ohmsma, Tokyo, 1990.

[76] Stephen Muggleton, Huma Lodhi, Ata Amini, and Michael J. E. Sternberg. Support vector inductive logic programming. In Achim Hoffmann, Hiroshi Motoda, and Tobias Scheffer, editors, *Proceedings of the 8th International Conference on Discovery Science*, volume 3735 of *LNCS*. Springer, 2005.

[77] Jennifer Neville, Brian Gallagher, and Tina Eliassi-Rad. Evaluating statistical tests for within-network classifiers of relational data. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, 2009.

[78] Jennifer Neville and David Jensen. Dependency networks for relational data. In *Proceedings of the IEEE International Conference on Data Mining*, 2004.

[79] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 809–816, 2011.

[80] Ontoprise. Neue Version des von Ontoprise entwickelten Ratgebersystems beschleunigt die Roboterwartung. *Ontoprise Pressemitteilung*, 2007.

[81] S. Peri, J. D Navarro, T. Z Kristiansen, R. Amanchy, V. Surendranath, B. Muthusamy, T. K. B. Gandhi, K. N. Chandrika, N. Deshpande, S. Suresh, et al. Human protein reference database as a discovery resource for proteomics. *Nucleic acids research*, 32(Database Issue):D497, 2004.

[82] A. Popescul and L. H Ungar. Feature generation and selection in multi-relational statistical learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[83] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3), 1990.

[84] Predrag Radivojac, Kang Peng, Wyatt T. Clark, Brandon J. Peters, Amrita Mohan, Sean M. Boyle, and Sean D. Mooney. An integrated approach to inferring gene-disease associations in humans. *Proteins*, 72, 2008.

[85] Luc De Raedt, Manfred Jaeger, Sau Dan Lee, and Heikki Mannila. A theory of inductive query answering. In *Proceedings of the IEEE International Conference on Data Mining*, 2002.

[86] S. Reckow and V. Tresp. Integrating ontological prior knowledge into relational learning. Technical report, Siemens, 2007.

[87] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *World Wide Web Conference*, 2010.

[88] A. Rettinger, M. Nickles, and V. Tresp. A statistical relational model for trust learning. In *Proceeding of 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, 2008.

[89] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[90] Celine Rouveirol and Veronique Ventos. Towards learning in CARIN-ALN. In *International Workshop on Inductive Logic Programming*, 2000.

[91] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR*, pages 43–50, 2005.

[92] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *NIPS*, pages 926–934, 2013.

[93] Steffen Staab and Andreas Hotho. Machine learning and the Semantic Web. In *ICML 2005 tutorial*, 2005.

[94] C. Stark, B. J Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. BioGRID: a general repository for interaction datasets. *Nucleic acids research*, 34(Database Issue):D535, 2006.

[95] Benjamin Taskar, Ming Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *Proceedings of Advances in Neural Information Processing Systems*, 2003.

[96] Volker Tresp, Markus Bundschus, Achim Rettinger, and Yi Huang. *Uncertainty Reasoning for the Semantic Web I*, chapter Towards Machine Learning on the Semantic Web. Lecture Notes in AI, Springer, 2008.

[97] Volker Tresp, Yi Huang, Markus Bundschus, and Achim Rettinger. Materializing and querying learned knowledge. In *Proceedings of the First ESWC Workshop on Inductive Reasoning and Machine Learning on the Semantic Web*, 2009.

[98] Volker Tresp, Yi Huang, Xueyan Jiang, and Achim Rettinger. Graphical models for relations - modeling relational context. In *KDIR 2011 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, Paris, France, 26-29 October, 2011*, pages 114–120, 2011.

[99] Volker Tresp and Kai Yu. Learning with dependencies between several response variables. In *Tutorial at the International Conference on Machine Learning*, 2009.

[100] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2071–2080, 2016.

[101] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966c.

[102] W. Van Laer and L. De Raedt. How to upgrade propositional learners to first order logic: A case study. In *Machine Learning and Its Applications, Advanced Lectures*, 2001.

[103] S. V. N. Vishwanathan, Nic Schraudolph, Risi Imre Kondor, and Karsten Borgwardt. Graph kernels. *Journal of Machine Learning Research - JMLR*, 2008.

[104] Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, 2001.

[105] Zhao Xu, Kristian Kersting, and Volker Tresp. Multi-relational learning with gaussian processes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009.

[106] Zhao Xu, Volker Tresp, Kai Yu, and Hans-Peter Kriegel. Infinite hidden relational models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006.

[107] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[108] Kai Yu, Wei Chu, Shipeng Yu, Volker Tresp, and Zhao Xu. Stochastic relational models for discriminative link prediction. In *Proceedings of Advances in Neural Information Processing Systems*, 2006.

[109] Xiaojin Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences TR 1530 University of Wisconsin Madison, 2006.