Structural Inference for Semistructured Data

Jason Sankey
Basser Department of Computer Science
University of Sydney
Sydney, NSW 2006, Australia
soda@cse.unsw.edu.au

Raymond K. Wong
School of Computer Science & Engineering
University of New South Wales
Sydney, NSW 2052, Australia
wong@cse.unsw.edu.au

ABSTRACT

Semistructured data presents many challenges, mainly due to its lack of a strict schema. These challenges are further magnified when large amounts of data are gathered from heterogeneous sources. We address this by investigation and development of methods to automatically infer structural information from example data. Using XML as a reference format, we approach the schema generation problem by application of inductive inference theory. In doing so, we review and extend results relating to the search spaces of grammatical inferences. We then adapt a method for evaluating the result of an inference process from computational linguistics. Further, we combine several inference algorithms, including both new techniques introduced by us and those from previous work. Comprehensive experimentation reveals our new hybrid method, based upon recently developed optimisation techniques, to be the most effective.

1. INTRODUCTION

Given the recent emergence of semistructured data, there are many problems that must be solved to facilitate its most effective use. Amongst the most important of these involves the ability to infer the structure of a semistructured data source to facilitate applications such as document recognition and categorisation, document validation, computer-assisted schema design, database browsing and query optimisation (such as Dataguides [10]). To design and implement an efficient structure inference method for semistructured data, one must need to able to compare the inferred structures produced by different methods or parameters.

Using XML as a reference format, this paper approaches the above schema generation problem by application of inductive inference theory. In doing so, we review and extend results relating to the search spaces of grammatical inferences. We then adapt a method for evaluating the result of an inference process from computational linguistics. This measure for the quality of an inferred DTD will be seen in section 3. Finally, we combine several inference algorithms, including both new techniques introduced by us and those from previous work. Comprehensive experimentation reveals our new hybrid method to be the most effective.

The inference of structure in semistructured data is a relatively new area of research. However, there are several closely related topics that have been studied for a longer period. Many of these topics fall into the general field of Inductive Inference, more specifically the sub-field of Grammatical Inference. This sub-field is concerned with the theory and methods for learning grammars from example data. For further details concerning the field of Grammatical Inference the reader is referred to the surveys of Pitt [13] and Sakakibara [16].

In addition to the work in the broader area of Grammatical Inference, there has also been prior research into automatic recognition of document structure. Earlier attempts by [5], [7] and [17] in similar problem spaces all use solutions based on heuristic methods. In each case, the generalisation step involves searching for similar patterns in the data and combining the corresponding structural information. Although these techniques may work well in some cases, their applicability is restricted by a lack of generality.

The approaches of [1] and [19] are more powerful in concept. In both of these works, methods derived from theoretical grammatical inference are applied to the problem of inferring DTD content models. The first known application of such theory to this problem, in [1], makes use of a characterising method to infer a subset of the regular language class. The alternative solution in [19] makes use of an adapted stochastic method. In both cases, the results are post-processed to produce more desirable content models. Unfortunately, neither paper investigates or compares other methods. It is partly for this reason that the methods they propose have been included in this study, and are further discussed in section 5.1.

The more recent paper of Garofalakis et al [8] is very similar to this work in terms of motivation and the application of information theory. However, the approach proposed in their DTD inference system, called XTRACT, is significantly different to that which we propose here. The inference algorithms of XTRACT are based upon direct generalisation and factoring of regular expressions, with information theoretic principles used to choose a final result from a pool of candidates. In contrast, we propose a method which employs similar principles throughout the inference process, with the aim of producing a more general method.

In addition to works concerned with the exact problem of DTD inference, there have also been many papers published on related topics. Most notable amongst these is work within the Lore [11] semistructured database project to infer Dataguides [10]. These are relatively simple, though useful, models of database structure.

The work is presented in the following fashion. Section 2 provides an overview of our solution method, followed by details of how we evaluate solutions in section 3. The fourth section gives an overview of our implementation, and we describe the important algorithms in section 5. Section 6 includes comprehensive testing and discussion, followed by conclusions in section 7.

2. OVERVIEW OF SOLUTION METHOD

The major grammatical inference methods fall into a few general categories. One of these categories includes a family of algorithms known as state merging methods.

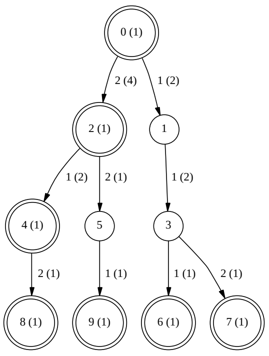

Figure 1: An example PTA

A state merging method typically begins by constructing what is known as a **Prefix Tree Automaton** (PTA) from the positive examples of the language to be inferred. If we let the set of positive examples be $R+$, then the prefix tree for $R+$ ($PTA(R+)$) may be constructed as follows. We begin with an automaton that simply accepts the first string in $R+$. Then we iterate over the rest of the strings in $R+$ and for each one follow transitions in the automaton for as many symbols as possible. When a symbol is found that does not match a valid transition, the PTA is augmented with a new path to accept the rest of the string. For example, given a set of input strings over the alphabet {1, 2} such as $R+ = \{\epsilon, 2, 21, 221, 212, 112, 111\}$, the PTA for $R+$ would be as shown in figure 1. The reader may notice the numbers in parentheses in each final state and for each transition. These represent the frequencies of the state finalities and transitions, allowing this type of PTA to be used as a **Probabilistic Finite State Automaton** (PFSA). A PFSA is merely an automaton with probabilities associated with each transition and final state. This is important both for some of the inference methods and for evaluating the quality of solutions.

The PTA represents a language that will accept exactly the positive examples and no other strings. From this the task is to generalise the PFSA to a model that will still accept the positive examples, along with others inferred to belong in the same language (the target language of the inference procedure.) Generalisation is performed by merging states of the automaton. When a pair of states are merged a new state is created, and all incoming and outgoing transitions to and from the original pair are redirected to and from the new state respectively. Choosing which state pairs to merge is dependent on the search order and an equivalence relation. Most methods in the state merging family are distinguishable by their equivalence relations, relatively few are dependent upon the searching order.

After a generalised PFSA is obtained further processing is dependent upon the application. For the production of a DTD, the disambiguation and conversion methods of [1] may be used. In other cases, particularly uses in semistructured databases, the PFSA would likely be converted into a more appropriate format (perhaps the native format of the database) but otherwise remain the same. For this reason the scope of this work is kept to producing the best possible generalisation in PFSA form. This form has the advantage of being the most general, particularly as it preserves statistical information.

3. EVALUATING A SOLUTION

Before attempting to automatically produce DTDs for given data, we need to establish how the quality of an inferred DTD is determined. Without such a method, there is no way to measure the success of an inference or indeed the viability of an inference method. The major problem in evaluating the utility of a DTD is the trade-off between the size of a content model and its expressive power.

The concept of Minimum Message Length (MML) was first formally introduced by Wallace and Boulton in [18]. Formally, the principle of MML states that the best abstraction of some given examples is the one which has the highest posterior probability. This is equivalent to maximising the product of the prior probability of the abstraction and probability of observing the examples given the abstraction. Now suppose we are given an abstraction (or *theory*) T with a prior probability $p(T)$ that attempts to explain some example data D. Then the posterior probability of T given D, denoted $p(T|D)$, is:

$$p(T|D) = \frac{p(TD)}{p(D)} = \frac{p(T)p(D|T)}{p(D)}$$

by applying Bayes' theorem. Thus to maximise $p(T|D)$ we must maximise $p(T)p(D|T)$. By the properties of logarithms and the known range of probabilities, it can be seen that this is equivalent to minimising $-\log p(T) - \log p(D|T)$. Further, from information theory it is known that the optimal code length for a symbol of probability p is $-\log p$. Thus maximising $p(T|D)$ is equivalent to minimising the data length required to encode it.

Given MML as the measure, we must establish a method of encoding PFSA. This problem was first addressed under the principle of MML in [9], and has since been corrected and developed in [15]. The formula is given below:

$$MML(A) = \sum_{j=1}^{N} \left\{ \log_2 \frac{(t_j - 1)!}{(m_j - 1)! \prod_{i=1}^{m_j} (n_{ij} - 1)!} \right\} + M(\log_2 V + 1) + M' \log_2 N - \log_2 (N-1)! \quad (1)$$

where N is the number of states in the PFSA, V is the cardinality of the alphabet plus one, t_j is the number of times the jth state is visited, m_j is the number of arcs from the jth state (plus one for final states), m'_j is the number of arcs from the jth state (no change for final states), n_{ij} is the frequency of the ith arc from the jth state, M is the sum of all m_j values and M' is the sum of all m'_j values.

4. IMPLEMENTATION OVERVIEW

One of the goals of this work was to both produce new inference methods and make comprehensive comparisons with existing techniques. The implementation consists of several modules to perform stages of the inference process. These act as pipelines to process from the XML data to a generalised PFSA, and to measure the quality of the inferred models. The most important stage involves PTA generalisation using the inference methods. These methods fall into two broad categories, which are labelled generalisation and optimisation in the implementation. The first of these consists of the Merge methods which roughly follow the merge idea described in section 2. The actual core of the implementation works closely to the pseudo-code in algorithm 1.

Algorithm 1 GeneralisePTA

```
Input: A PTA $A$ to be generalised, a merge criterion $criterion$
Output: The generalised form of $A$
Method:
 1. repeat
 2.   for all pairs $(s_1, s_2)$ of states in $A$ do
 3.     if $criterion(s_1, s_2)$ then
 4.       $A.merge(s_1, s_2)$
 5.       $criterion.forcedMerges(A)$
 6.       if $criterion.determinise()$ then
 7.         determinise($A$)
 8.       end if
 9.     end if
10.   end for
11. until no more merges are possible
12. if not $criterion.determinise()$ then
13.   determinise($A$)
14. end if
15. return $A$
```

Here the merge criterion determines the actual behaviour of the inference procedure. For each method belonging to the merge family, a merge criterion class is derived from a base interface. The merge criterion is allowed to make forced merges after an initial merge is decided (see line 5), which may be necessary to fit the semantics of a method, or may be more efficient. Also, merge criteria may decide if the PFSA is determinised after every merge (lines 6–8) or only at the end of the inference process (lines 12–14). Determinisation itself is performed by merging of states.

The alternative inference methods all apply optimisation techniques to try and minimise the MML of the PTA. These algorithms vary significantly in implementation, ruling out the possibility of building them around the same core algorithm.

5. INFERENCE ALGORITHMS

5.1 Reference Methods

To provide a comparison of several possible inference methods, several previously applied methods were evaluated. Due to the number of algorithms implemented, full descriptions of them are not presented here. Instead, we provide references to relevant literature for the interested reader.

Two such algorithms are those applied by Ahonen in [1], the first known paper to address DTD generation using tradition grammatical inference methods. These methods are theoretically appealing, as they guarantee to infer languages falling within certain language classes. These classes are termed k-contextual and (k, h)-contextual, so named as they assume the structure to be inferred to have limited context. It is not clear whether this assumption is valid in practice, however. Thus the theoretical appeal may not necessarily translate into practical applicability.

Another method applied to DTD generation ([19]), is derived from more recent work in grammatical inference. The base algorithm is known as Alergia, introduced in [4]. The criterion for state equivalence in Alergia is based upon observed frequencies of transitions and finalities of a pair of states. As with the methods of Ahonen, Alergia has strong theoretical appeal. Again, though, we are interested in practical performance.

Further to the methods described above, we devised two basic optimisation strategies against which to benchmark the results of our main algorithm. The first of these, termed the Greedy method, is a straightforward steepest-descent algorithm which employs incremental MML calculation to optimise a PTA. Along with this a weighted stochastic hill-climbing method was implemented, which also used incremental MML calculation. These two methods illustrate the need for more sophisticated optimisation algorithms.

5.2 The sk-strings Method

The sk-strings method actually consists of a family of algorithms, described in [14], of which five were implemented. The basis of these algorithms is an extension upon the k-tails heuristic of Biermann and Feldman [2], which in turn is a relaxed variant of the Nerode equivalence relation. Under the Nerode relation, a pair of states are equivalent if they are indistinguishable in the strings that may be accepted following them. The k-tails criterion relaxes this to only considering these strings (tails) up to a given length (k.) The sk-strings method is extended using stochastic automata, and considers only the top s percent of the most probable k-strings. The k-strings differ from k-tails in that they are not required to end at a final state, unless they have length less than k. The probability of a k-string is calculated by taking the product of the probabilities of the transitions exercised by that k-string.

Of the five variants implemented, the first four are quite similar, they differ only in the strictness of the equivalence measure. These four variants are the **OR**, **AND**, **LAX** and **STRICT** heuristics. The final variant implemented is quite different to these four, in that it uses a cross-entropy measure to evaluate if two states are equivalent. The divergence between the distributions of k-strings from the two states is calculated, and they are deemed to be equivalent only if the result falls below a threshold value. This heuristic has been labelled **XEN** (short for cross-entropic), and the details are presented in [15].

Although there is not space to present each of the variants here, the pseudo-code of algorithm 2 provides an example implementation for the **AND** heuristic. Raman provides greater detail on each of the variants in [15], including significant experimental results. However, the available source

Algorithm 2 andCriterion

Input: A real number s in [0, 1], a positive integer k, and a pair of states s_1 and s_2.

Output: A boolean, true iff s_1 and s_2 are sk-AND equivalent.

Method:

```
tails ← k-strings of $s_1$, sorted by decreasing order of probability
total ← 0
for tail in tails do
    total ← total + tail.probability
    if $\delta(s_2, tail.string) = \emptyset$ then
        return false
    end if
    if total ≥ s then
        tails ← k-strings of $s_2$, sorted by decreasing order of probability
        total ← 0
        for tail in tails do
            total ← total + tail.probability
            if $\delta(s_1, tail.string) = \emptyset$ then
                return false
            end if
            if total ≥ s then
                return true
            end if
        end for
        return true
    end if
end for
return false
```

code for the implementation contains a serious bug in the calculation of MML values, so it is unknown if the results are reliable. Regardless, we present independent tests in section 6.

5.3 Ant Colony Optimisation

The Ant Colony Optimisation (ACO) meta-heuristic is a relatively new optimisation technique. First described in [6], the method uses a positive feedback technique for searching in a similar manner to actual ants. It operates over several iterations to allow the positive feedback of the pheromones to take effect. In each iteration, the artificial ants navigate the search space using only a simple heuristic, but as they move they leave pheromones on the trail they follow. In some variants, including the one used in this work, the pheromone placement is delayed until the end of an iteration, when all ants have completed a full walk. At this point the amount of pheromone assigned to each ant is weighted with respect to the quality of the solution it found. Thus moves involved in higher quality solutions are more likely to be chosen by ants in future iterations. Although ants acting by themselves are only capable of finding relatively poor solutions, working in cooperation they may approach higher quality ones. After a certain number of iterations without improvement to the best solution found the algorithm terminates. This process is shown in pseudo-code in algorithm 3.

Notable in algorithm 3 is the mention of an *ant move selector*. This is an input to the algorithm allowing customisation of the way in which ants will select state pairs to merge. An ant move selector consists of two parts, an *ant heuristic* and an *ant weighting function*. The former is used to calculate a heuristic value for a given state merge. The latter is a function to combine this heuristic value with the pheromone value associated with the merge. Given both of these measurements, an ant move selector will choose a merge as outlined in algorithm 4. On line 8 of the algorithm, a function *stochasticChoice* is used to randomly select a move based upon the values derived at line 5. Each merge is accorded a probability of its own value over the total. This allows some random exploration of the search space, with bias towards merges with higher heuristic and pheromone values. Each ant employs the move selector to repeatedly choose merges until the ant dies. Death occurs when either the automaton is reduced to one state, or the search is abandoned due to all merges having a negative heuristic value. After its death, an ant keeps stored the best solution found in its walk of the lattice.

Algorithm 3 ACOOptimiser

Input: A PTA PTA_{R+}, a positive integer *antCount*, a positive integer *stagLimit*, and an ant move selector *ams*.

Output: A generalised form of the input PTA.

Method:

```
bestSolution ← $PTA_{R+}$
stagCount ← 0
while stagCount < stagLimit do
    ants ← makeAnts(antCount, $PTA_{R+}$, ams)
    liveCount ← antCount
    improvement ← false
    while liveCount > 0 do
        for ant in ants do
            ant.step()
            if ant.dead() then
                if mmlOf(ant.solution) < mmlOf(bestSolution) then
                    bestSolution ← ant.solution
                    improvement ← true
                end if
                liveCount ← liveCount − 1
            end if
        end for
    end while
    updatePheremones(ants)
    if not improvement then
        stagCount ← stagCount + 1
    else
        stagCount ← 0
    end if
end while
return bestSolution
```

The heuristic measure used by an ant move selector may be varied, however only one such heuristic was implemented. This heuristic evaluates a merge based upon the MML change that would result. The actual formula used to evaluate a state pair *merge* in a PFSA A is:

$$h = \frac{-\delta MML(A, merge) + MML(A)}{MML(A)}$$

Algorithm 4 antMoveSelector

Input: A set of all state pairs *merges*, an ant heuristic *heuristic*, an ant weighting function *weighting* and a pheromone table *pheremones*.

Output: A state pair representing the chosen merge.

Method:

```
choices ← [ ]
for $merge$ in $merges$ do
    $h \leftarrow heuristic(merge)$
    $p \leftarrow pheremones[merge]$
    $value \leftarrow weighting(h, p)$
    $choices.add((value, merge))$
end for
return stochasticChoice($choices$)
```

where $\delta MML(A, merge)$ is used to represent the change in the MML of A that would result from *merge*. The weighting function used to combine the heuristic and pheromone values may also be varied, though again only one function was implemented. This function has two variables, α and β, which are used as exponential weights for the pheromone and heuristic values respectively. Thus the weighted combination of a pheromone level p and a heuristic value h would be:

$$value = p^{\alpha} + h^{\beta}$$

The final detail of the algorithm is the manner in which pheromones are updated. As mentioned earlier, the laying of pheromone trails is delayed until the end of an iteration to allow the pheromone values to be weighted. The weighting applied is based upon an individual ant's performance compared with the average for the iteration. For an ant a, the accorded pheromone is thus:

$$p = \frac{averageMML}{mmlOf(a.solution)}$$

where $averageMML$ is the average for all ants in that iteration. This amount of pheromone is laid for every state pair used in the construction of the ant's solution. As most merges will contain states formed from prior merges, care must be taken to assign the pheromones to all combinations of the original states involved.

5.4 Our Proposed Hybrid Method

Initial testing of the kind presented in section 6 revealed that the merge family methods performed well in some cases, and the optimisation methods better in others. This led to the development of a new algorithm, based on the most successful method of each type.

The sk-ANT Heuristic: The motivation for this new heuristic was to create a method that would be successful for a variety of input data sizes by combining the best features of both the sk-strings and ACO techniques. One consideration was to first run the sk-strings algorithms, and then use the results to seed the ACO optimisation. However, this approach suffers from several problems. Firstly, it is not practical to attempt all possible combinations of both algorithms. Thus we would be required to choose a limited number of models resulting from the sk-strings technique to seed the second stage of the process. The simplest way to achieve this would be to choose the best models, up to a reasonable number. These models will not necessarily lead to the best results, though, as they may have already been over-generalised. More importantly, by letting the sk-strings methods run to completion we would lose many of the advantageous aspects of the ACO method. Most notably, its willingness to explore a greater breadth of the search space would be missed.

The new method thus incorporates both the sk-strings and ACO heuristics at each step of the inference process. It is most easily described as a modified version of the ACO technique with the ants guided by an sk-strings criterion. The guiding is made progressively weaker as the model becomes smaller, to allow the advantages of the ACO method for smaller models to take effect. The main modification was made to the ant move selection of algorithm 4, producing a new method as shown in algorithm 5. The first major difference appears on line 4, where a merge must pass the sk-strings criterion to be considered. The outer while loop on line 2 and if statement on line 11 combine to progressively weaken the sk-strings criterion when it has become too strict. Eventually the criterion will be weak enough to let all merges pass, and the algorithm will behave identically to the original version.

Algorithm 5 sk-antMoveSelector

Input: A set of all state pairs *merges*, an ant heuristic *heuristic*, an ant weighting function *weighting*, a pheromone table *pheremones* and an sk-strings criterion *skCriterion*.

Output: A state pair representing the chosen merge.

Method:

```
1.  choices ← [ ]
2.  while $choices.size() = 0$ do
3.      for $merge$ in $merges$ do
4.          if $skCriterion(merge)$ then
5.              $h \leftarrow heuristic(merge)$
6.              $p \leftarrow pheremones[merge]$
7.              $value \leftarrow weighting(h, p)$
8.              $choices.add((value, merge))$
9.          end if
10.     end for
11.     if $choices.size() = 0$ then
12.         $skCriterion.weaken()$
13.     end if
14. end while
15. return stochasticChoice($choices$)
```

6. EXPERIMENTAL RESULTS

To fulfill our goal of comprehensive testing, we applied three sets of tests. The first two of these used generated data, which allowed systematic experimentation on widely varied input. The other test set consisted of a few models chosen from real data, to illustrate the nature of the models inferred by the sk-ANT method. In each test the algorithms were run with a range of input parameters, and the results from each run combined.

6.1 Inference of Small Models

The smaller generated data set consisted of 100 sample files generated from random PFSA with a maximum of 5 states and an alphabet cardinality of 4. A total of 10 sample strings were generated for each PFSA, leading to PTA with an average size of 42.15 states. The number of strings generated was deliberately kept small, as sparse data is an important problem in practical cases of inference. The average size of the models inferred by the algorithms ranged from 1.03 to 40.55 states, with the best models in terms of MML typically having between 2 and 5 states.

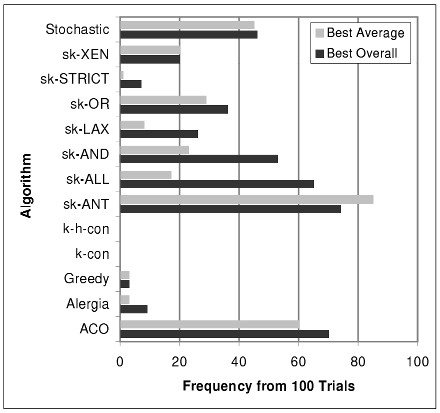

Figure 2: Success rates for small models

Figure 2 shows the success rates of each algorithm in inferring models with the lowest MML values. For each algorithm, two results have been shown. The first is the frequency of inferring the best model overall, by choosing the best of the algorithm's attempts. The second is the frequency of obtaining the best average performance, derived by averaging all of the algorithm's attempts before ranking. The best overall performance is most important, whilst the best average indicates stability across diferring input parameters. The results clearly show that the sk-ANT algorithm performed best in terms of both rankings, particularly the best average ranking. The next two best algorithms were the original ACO method, and the sk-ALL heuristic (a combination of the results from all ak-strings variants.) This is one of the reasons those methods were chosen as the basis for sk-ANT. The other previously applied algorithms and reference methods performed quite poorly. Although the Stochastic method was able to infer some good models, it trailed significantly.

Algorithm	Average Deviation (%)	Worst Deviation (%)
ACO	3.41	41.50
Alergia	16.06	62.25
Greedy	91.45	459.04
k-contextual	19.43	53.43
(k, h)-contextual	17.11	45.39
sk-ANT	0.91	15.20
sk-ALL	1.69	17.50
Stochastic	5.53	45.01

Table 1: Deviation from the best model inferred (small data set)

Statistics relating to the consistency of the algorithms over the 100 test cases were also gathered, in the form of comparisons against the best inferred models. Table 6.1 shows both the average deviation and worst deviation in percent for each of the algorithms. The numbers were derived from the difference between the MML values of the best model inferred by a given algorithm as compared with the best model overall. We omit the individual sk-strings algorithms, preferring the combined sk-ALL results. The results show that the sk-ANT method is the best in terms of average and worst deviation, followed by the sk-ALL hueristic. Note that the worst case deviations may be too high for some applications. In such cases, using a combined approach with both the sk-ANT and sk-ALL algorithms would yield more consistent results.

6.2 Inference of Large Models

The larger data set also consisted of 100 sample files generated from random PFSA. In this case the PFSA had a maximum of 20 states with an alphabet cardinality of 8. For each of these a total of 25 strings were generated giving an average PTA size of 143.38 states. On average the inferred models were larger than for the small data set, though they ranged from an average of 1.23 to 142.74 states for differing algorithms.

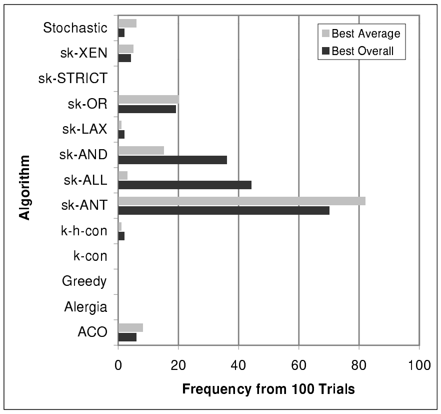

Figure 3: Success rates for large models

Figure 3 shows the relatively success rates for the algorithms, derived as they were for small models. The largest change in the results is the much poorer performance of both the Stochastic and ACO methods, and the clear dominance of the sk-ANT heuristic. The reduction in performance of the ACO and Stochastic methods is likely due to the vastly increased size of the search space. The heuristic guidance used in the sk-ANT method clearly overcomes this difficulty, producing the best results. Finally, the algorithms which performed poorly on the small data sets suffered a similar fate on the larger models. These algorithms are desirable due to their simplicity and efficiency, but are lacking in quality of solutions found.

Again the deviations from the best model were calculated for each algorithm, and are presented in table 6.2. The hybrid sk-ANT technique again proved to be the best in terms of average deviation at 2.81%. Thus the newer method is preferable if only one choice is allowed. On the other hand, the sk-ALL heuristic was better in terms of worst-case performance, though the deviation is still rather high. Again, some applications may need to employ a combination of difference methods to achieve better worst-case results.

Algorithm	Average Deviation (%)	Worst Deviation (%)
ACO	31.57	154.87
Alergia	32.44	86.58
Greedy	173.46	574.48
k-contextual	30.39	84.15
(k, h)-contextual	21.57	58.57
sk-ANT	2.81	28.52
sk-ALL	3.15	20.51
Stochastic	36.40	159.11

Table 2: Deviation from the best model inferred (large data set)

6.3 Discussion

The experiments have revealed many interesting points. Firstly, the k-contextual, (k, h)-contextual and Alergia methods previously applied to this problem have been shown to perform poorly. Although the papers describing their use extend the algorithms and employ refining operations, it appears that other methods are a more appropriate starting point. We have also seen that the simple Greedy and Stochastic methods cannot match the performance of more complicated techniques. This highlights the difficulties inherent in the search space of grammatical inference. The sk-strings method developed in [15] has proven to be much more effective, provided combined results from all of the heuristics are used. A major contribution to its success may lie in its use of statistical data in the inference process. We have also seen that the ACO method introduced in this work performs well on small models. The algorithm's subsequent failure on large testing data led to a new method which we have named sk-ANT. This hybrid algorithm proved to be the most effective by a considerable margin, and is thus the algorithm of choice. Where worst case guarantees are required, we recommend using combined results of both the sk-ANT and sk-ALL methods.

6.4 Real Data Sets

Rather than rely on results from artificial data alone, testing was also performed on real documents. Unfortunately, it is difficult both to obtain suitable data and to perform widespread tests with such a variety of algorithms. Thus this section is used only to present some samples of the best models inferred by the sk-ANT method. These results illustrate that models with low MML values do indeed capture important structural information.

6.4.1 Shakespeare SPEECH Element

The first of the real data sets used was a marked up form of Shakespeare's Othello. From this data set we chose the SPEECH element as the most interesting in terms of inferring a content model. The element was chosen due to the fact that many samples were present in the document, yielding a large PTA. Although the structure of a speech is fairly basic, the challenge was to uncover this simplicity from a large input automaton. Figure 4 shows the best model inferred by the sk-ANT method for the SPEECH element. This model identifies many major structural elements: all speeches begin with one or more speakers; a great deal of speeches are only a single line; a speech consists of lines mixed with stage directions; a speech always ends with a line; and a speech has at most two stage directions in a row.

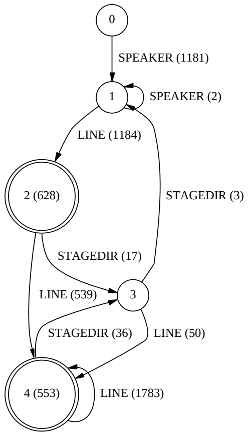

Figure 4: Best model for the Shakespeare SPEECH element

6.4.2 How-to SECT1 Element

The second of the real data sets used was derived from the How-to documents which form part of the Linux Documentation Project (LDP). This is a large collection of documentation available in several formats including SGML. A sample was taken and converted to XML format for the experiment. From this sample, we have chosen to present the results for an element tagged 'SECT1' here. This element forms part of a hierarchical method of dividing a document into sections, sub-sections and so on. This particular elements was chosen as it posses a strict ordering policy, mixed with allowance for repeated elements. As before, we show the best model inferred by the sk-ANT method in figure 5. This model identifies the following structural properties: all sections begin with a title; the remainder of a section consists of paragraphs and nested sections; nested sections occur after at most one paragraph; and nested sections are never followed by paragraphs.

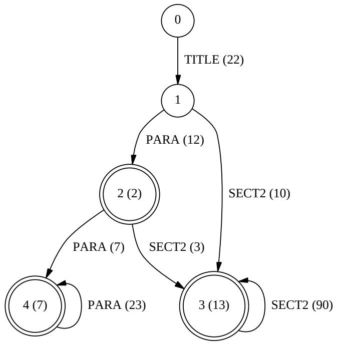

Figure 5: Best model for the SECT1 element

6.4.3 Webster's p Element

The final real data set was an extract of the digital form of the 1913 Webster Unabridged Dictionary. The dictionary format is almost compatible with SGML, and after some preprocessing we were able to extract the structural information in XML format. Only a small subset of the entire dictionary was used, due to its overwhelming size. We selected the paragraph element 'p', which is used to group together the information pertaining to each dictionary word, for the experiment. This particular element was chosen as it exhibits a rather complex and variable structure. This is in contrast to the structure of the other models chosen from real data sets, with the intent being to stretch the capabilities of the sk-ANT method.

The model shown in figure 6 is clearly more complex than the other two. However, observe the dominant path through states 0, 2, 9, 4 and 1. It is most important that this path is preserved, with the rest of the structure accounting for exceptions and noise in the data. Such irregularity is a fact of life when dealing with semistructured data.

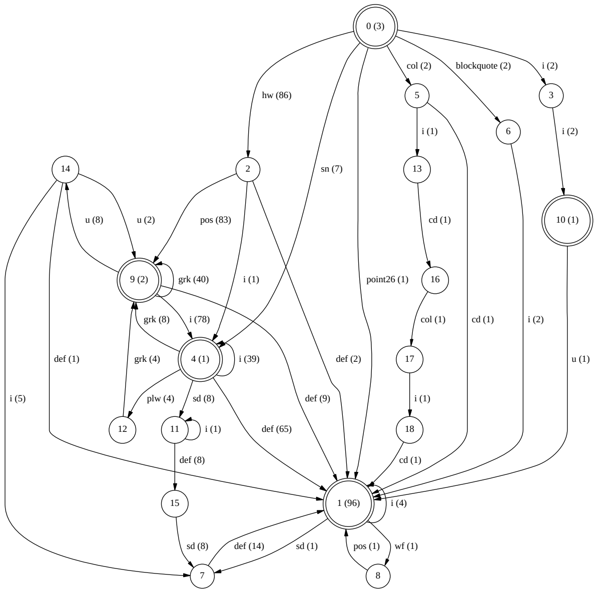

Figure 6: Best model for the ‘p’ element

7. CONCLUSION

We have addressed the problem of structural inference for semistructured data by investigating the particular problem of inferring Document Type Definitions from XML data. In doing so we began by motivating the research and identifying major applications. The use of Minimum Message Length as a measure for the quality of inferred content models has been introduced, adapted from work in related fields. This measure has proven to be an appropriate and a vast improvement over previous subjective techniques.

We have also presented the first wide spread comparison of different grammatical inference techniques. This involved the implementation of the existing Alergia, k-contextual, (k, h)-contextual and sk-strings methods, as well as the creation of new algorithms. The new methods include the Greedy strategy, the ACO meta-heuristic, Stochastic Hill Climbing and our proposed, hybrid sk-ANT heuristic. Comprehensive experimental data revealed that our proposed method was the most effective and most stable of the methods, followed by the sk-ALL heuristic. From this work we may conclude that the problem of structural inference is both important and tractable. For current applications we recommend use of the sk-ANT technique, possibly combined with the sk-ALL heuristic. The use of MML as a quality measure is also recommended, due to its generality and objectivity.

8. REFERENCES

[1] H Ahonen. *Generating Grammars for Structured Documents Using Grammatical Inference Methods.* Report A-1996-4, Department of Computer Science, University of Finland, 1996.

[2] A W Biermann and J A Feldman. *On the synthesis of finite-state machines from samples of their behaviour.* IEEE Transactions on Computers, 21:591–597, 1972.

[3] R C Carrasco and J Oncina (editors). *Grammatical Inference and Applications.* Proceedings of the Second International Colloquium on Grammatical Inference (ICGI-94), Lecture Notes in Artificial Intelligence 862, Springer-Verlag 1994.

[4] R C Carrasco and J Oncina. *Learning Stochastic Regular Grammars by Means of a State Merging Method.* In R C Carrasco and J Oncina (editors) [3].

[5] J Chen. *Grammar Generation and Query Processing for Text Databases.* Research proposal, University of Waterloo, January 1991.

[6] M Dorigo, V Maniezzo and A Coloni. *Positive Feedback as a Search Strategy.* Technical Report 91–016, Dipartmento di Elettronica, Politecnico di Milano, Italy, 1991.

[7] P Fankhauser and Y Xu. *Markitup! An Incremental Approach to Document Structure Recognition.* Electronic Publishing - Origination, Dissemination and Design, 6(4):447–456, 1994.

[8] M Garofalakis, A Gionis, R Rastogi, S Seshadri and K Shim. *XTRACT: A System for Extracting Document Type Descriptors from XML Documents.* In Proceedings of SIGMOD 2000, pages 165–176, Dallas, TX, 2000.

[9] M P Georgeff and C S Wallace. *A General Selection Criterion for Inductive Inference.* In T O'Shea (editor), ECAI-84: Advances in Artificial Intelligence, pages 473–481. Dordretch: Elsevier, 1984.

[10] R Goldman and J Widom. *Dataguides: Enabling Query Formulation and Optimization in Semistructured Databases.* In Proceedings of the Twenty-Third International Conference on Very Large Databases, pages 436–445, August 1997.

[11] J McHugh, S Abiteboul, R Goldman, D Quass and J Widom. *Lore: A database management system for semistructured data.* SIGMOD Record, 26(3):54-66, September 1997.

[12] S Nestorov, S Abiteboul and R Motwani. *Inferring Structure in Semistructured Data.* In Proceedings of the Workshop on Management of Semistructured Data (in Conjunction with PODS/SIGMOD), May 1997.

[13] L Pitt. *Inductive Inference, DFA's and Computational Complexity.* In J Siekmann (editor), Proceedings of the International Workshop AII '89, Lecture Notes in Artificial Intelligence 397, pages 18–44, Springer-Verlag 1989.

[14] A V Raman and J D Patrick. *The sk-strings method for inferring PFSA.* In Proceedings of the 14th International Conference on Machine Learning, ICML'97, 1997.

[15] A V Raman. *An Information Theoretic Approach to Language Relatedness.* PhD Thesis, Massey University, 1997.

[16] Y Sakakibara. *Recent Advances in Grammatical Inference.* Theoretical Computer Science Volume 185, Number 1, October 1997.

[17] K Shafer. *Creating DTDs via the GB-engine and Fred.* Available at **http://www.oclc.org/fred/**, 1995.

[18] C S Wallace and D M Boulton. *An Information Measure for Classification.* Computer Journal 11:185–194, 1968.

[19] M D Young-Lai. *Application of a Stochastic Grammatical Inference Method to Text Structure.* Master's thesis, Computer Science Department, University of Waterloo, 1996.