Machine Learning, 40, 111–137, 2000
© 2000 Kluwer Academic Publishers. Printed in The Netherlands.

Stochastic Grammatical Inference of Text Database Structure

MATTHEW YOUNG-LAI mdyounglai@neumann.uwaterloo.ca
FRANK WM. TOMPA
Computer Science Department, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1

Editor: Raymond Mooney

Abstract. For a document collection in which structural elements are identified with markup, it is often necessary to construct a grammar retrospectively that constrains element nesting and ordering. This has been addressed by others as an application of grammatical inference. We describe an approach based on *stochastic* grammatical inference which scales more naturally to large data sets and produces models with richer semantics. We adopt an algorithm that produces stochastic finite automata and describe modifications that enable better interactive control of results. Our experimental evaluation uses four document collections with varying structure.

Keywords: stochastic grammatical inference, text database structure

1. Introduction

1.1. *Text structure*

For electronically stored text, there are well known advantages to identifying structural elements (e.g., chapters, titles, paragraphs, footnotes) with *descriptive markup* (André, Furuta, & Quint, 1989; Berg, 1989; Coombs, Renear, & DeRose, 1987). Most commonly, markup is in the form of labeled tags interleaved with the text as in the following example:

```
<quotation>
<reference>The Art of War: Chapter 3 paragraph 18</reference>
<sentence>
If you know the enemy and know yourself, you need not fear the
result of a hundred battles. </sentence>
<sentence>
If you know yourself but not the enemy, for every victory gained
you will also suffer a defeat. </sentence>
<sentence>
If you know neither the enemy nor yourself, you will succumb in
every battle. </sentence>
</quotation>
```

Documents marked up in this way can be updated and interpreted much more robustly than if the structural elements are identified with codes specific to a particular system or typesetting style. The markup can also be used to support operations such as searches that require words or phrases to occur within particular elements.

Further advantages can be gained by using a formal grammar to specify how and where elements can be used. The above example, for instance, conforms to the following grammar represented as a regular expression:

$$\text{quotation} \rightarrow \text{reference sentence}^{+}.$$

This specifies that a quotation must contain a reference followed by one or more sentences. Any other use of the elements, e.g. nesting a sentence in a reference or preceding a reference by a sentence, is disallowed. Thus the main benefit of having a grammar is that new or modified documents can be automatically verified for compliance with the specification. Other benefits include support for queries with structural conditions and optimization of physical database layout. Another important purpose of a grammar is to provide users with a general understanding of the text's organization. Overall, a grammar for a text database serves much the same purpose as a schema for a traditional database: it gives an overall description of how the data is organized (Gonnet and Tompa, 1987).

The most widely used standard for text element markup and grammar specification is SGML (Standard Generalized Markup Language) (ISO, 1986), and more recently, XML (Bray, Paoli, & Sperberg-McQueen, 1998). HTML represents a specific application of SGML, i.e., it uses a single grammar and set of elements (although the grammar is not very restrictive).

1.2. Automated document recognition

Unfortunately, many electronic documents exist with elements and grammar only implicitly defined, typically through layout or typesetting conventions. For example, on the World Wide Web, data is laid out according to conventions that must be inferred to use it as easily as if it were organized in a database (Suciu, 1997). There is therefore a pressing need to convert the structural information in such documents to a more explicit form in order to gain the full benefits of their online availability. Completely manual conversion would be too time consuming for any collection larger than a few kilobytes. Therefore, automated or interactive methods are needed for two distinct sub-problems: element recognition and grammar generation.

If the document is represented by procedural or presentational markup, the first sub-problem is to recognize and mark up individual structural elements based on layout or typesetting clues. To do this, it is necessary to know or infer the original conventions used to map element types to their layout. We do not address this task here, but there are several existing approaches, based on interactive systems (Fankhauser & Xu, 1993; Kilpeläinen et al., 1990), on learning systems (Muslea, Minton, & Knoblock, 1998), and on manual trial and error construction of finite state transducers (Clark, 1991; Kazman, 1986).

The second sub-problem is to extract implicit structural rules from a collection of documents and model them with a grammar. This requires that a plausible model of the original intentions of the authors be reconstructed by extrapolating from available examples in some appropriate way. This can be considered an application of *grammatical inference*—the general problem that deals with constructing grammars consistent with training data (Vidal, 1994).

Note that the two problems may depend on each other. Element recognition often requires that ambiguities be resolved by considering how elements are used in context. However, recognition usually considers these usage rules in isolation and identifies only the ones that are really needed to recognize an element. A grammar can be considered a single representation of all usage rules, including the ones that are not relevant to recognition (which may be the majority). Thus, even if certain components of the grammar need to be determined manually in the recognition phase, grammar inference is still useful for automatically combining these components and filling in the ones that were not needed for recognition.

The grammar inference problem is especially applicable to XML, an increasingly important variant of SGML in which rich tagging is allowed without requiring a DTD (grammar). In this case, there is no recognition subproblem and grammar generation comprises the entire recognition problem.

The benefits of attaching a grammar to documents can be seen from the recent experience with the database system Lore (McHugh et al., 1997). Lore manages semistructured data, where relationships among elements are unconstrained. In place of schemas, "DataGuides" are automatically generated to capture in a concise form the relationships that occur (Goldman & Widom, 1997). These are then used for traditional schema roles such as query optimization and aiding in query formulation.

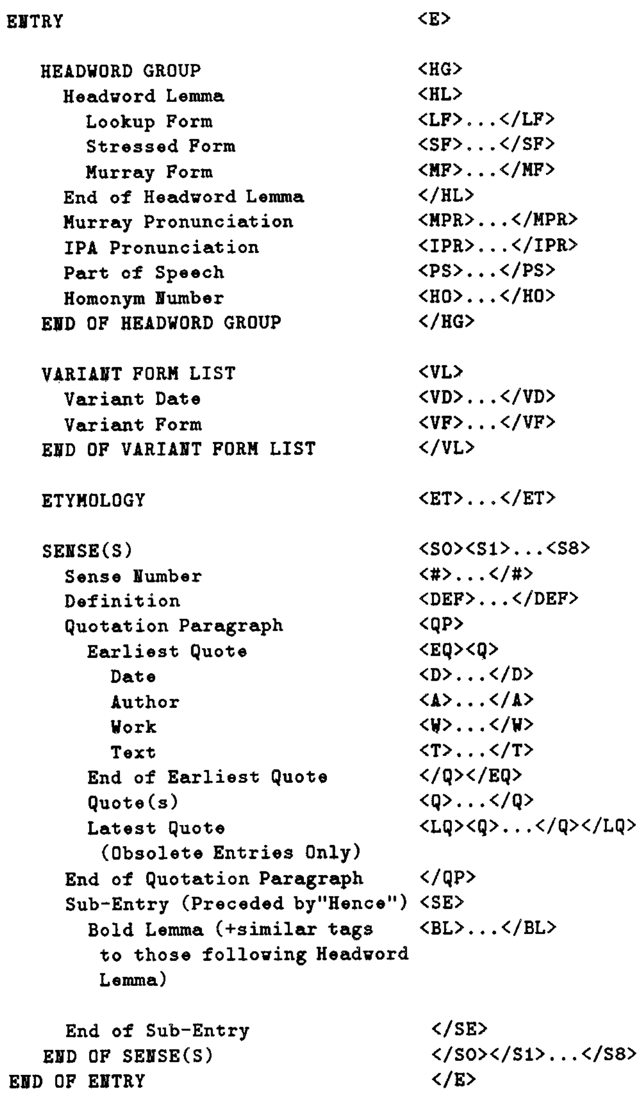
1.3. The data

We describe our approach to this problem using the computerized version of the *Oxford English Dictionary* (OED) (Simpson & Weiner, 1989). This is a large document with complex structure (Oxford University Press, 1998), containing over sixty types of elements that are heavily nested in over two million element instances. Figure 1 lists some of the most common elements as they appear in a typical entry. See the guide by Berg (1993) for a complete explanation of the structural elements used in the OED.

The OED has been converted from its original printed form, through an intermediate keyed form with typesetting information, to a computerized form with explicitly tagged structural elements (Kazman, 1986). The text is broken up into over two hundred ninety thousand top level elements (dictionary entries). The grammar inference problem can be considered as one of finding a complete grammar to describe these top level entries, or it can be broken into many subproblems of finding grammars for lower level elements such as quotation paragraphs or etymologies.

As a choice of data, the OED is a good representative of the general problem of inferring grammars for text structure. It is at least as complex as any text in two ways: the amount of structure information is extremely high, and the usage of structural elements is quite variable.

The large amount of structure information is evidenced by the number of different types of tags, and also by the high density of tagging as compared to most text (Raymond, 1993). There are over sixty types of tags, and the density of tagging is such that, even with all



ENTRY	`<E>`
HEADWORD GROUP	`<HG>`
Headword Lemma	`<HL>`
Lookup Form	`<LF>...</LF>`
Stressed Form	`<SF>...</SF>`
Murray Form	`<MF>...</MF>`
End of Headword Lemma	`</HL>`
Murray Pronunciation	`<MPR>...</MPR>`
IPA Pronunciation	`<IPR>...</IPR>`
Part of Speech	`<PS>...</PS>`
Homonym Number	`<HO>...</HO>`
END OF HEADWORD GROUP	`</HG>`
VARIANT FORM LIST	`<VL>`
Variant Date	`<VD>...</VD>`
Variant Form	`<VF>...</VF>`
END OF VARIANT FORM LIST	`</VL>`
ETYMOLOGY	`<ET>...</ET>`
SENSE(S)	`<S0><S1>...<S8>`
Sense Number	`<#>...</#>`
Definition	`<DEF>...</DEF>`
Quotation Paragraph	`<QP>`
Earliest Quote	`<EQ><Q>`
Date	`<D>...</D>`
Author	`<A>...</A>`
Work	`<W>...</W>`
Text	`<T>...</T>`
End of Earliest Quote	`</Q></EQ>`
Quote(s)	`<Q>...</Q>`
Latest Quote (Obsolete Entries Only)	`<LQ><Q>...</Q></LQ>`
End of Quotation Paragraph	`</QP>`
Sub-Entry (Preceded by"Hence")	`<SE>`
Bold Lemma (+similar tags to those following Headword Lemma)	`<BL>...</BL>`
End of Sub-Entry	`</SE>`
END OF SENSE(S)	`</S0></S1>...</S8>`
END OF ENTRY	`</E>`

Figure 1. Common elements from the OED with their corresponding tags in the order they appear in a typical dictionary entry. (This is reproduced from Berg (1989) and represents a human-generated high level description of the data.) Indentation denotes containment. Therefore, the headword group, variant form list, etymology, and senses are top level elements of an entry; their children are indented one step further; etc. Note that more than one sense element can occur at the top level.

element names abbreviated to three or fewer characters, the amount of tagging is comparable to the amount of text.

In most text, restrictions on element usage tend to vary between one extreme where any element can occur anywhere (e.g. italics in HTML), to the other where elements always occur in exactly the same way (e.g. a newspaper article which always contains a title, followed by a byline, followed by a body). Neither of these extremes is interesting for grammar inference. Element usage in the OED, however, is constrained, yet quite variable. This is mainly a consequence of the circumstances of its original publication. The compilation spanned 44 years, and filled over 15,000 pages. There were seven different chief editors over the lifetime of the project, most of which was completed before 1933—well before the possibility of computer support. The structure of the OED is remarkably consistent, but variable enough to be appropriate for the problem: it can test the method but is regular enough to make description by a grammar appropriate.

1.4. *Text structure and grammatical inference*

We now demonstrate the use of marked up text from the OED as training data for a grammatical inference process. Consider the structure of the two short OED entries shown in figure 2. These can be represented by the derivation trees, shown in figure 3, where the nodes are labeled with their corresponding tag names. A corresponding grammar representation is shown in figure 4. Each production has a left hand side corresponding to a non-leaf node, and a right hand side formed from its immediate children. The number of times a

```
<E><HG><HL><LF>salamandrous</LF><SF>sala&sm.mandrous</SF>
<MF>salama&sd.ndrous </MF></HL><b>,</b> <PS>a.</PS></HG>
<LB>rare</LB><su>-1</su>. <ET>f. <L> L.</L> <CF>
salamandra</CF> <XR><XL>salamander</XL></XR> + <XR>
<XL>-ous</XL></XR>.</ET> <S4><S6><DEF>Living as it were
in fire; fiery, hot, passionate. </DEF><QP><Q><D>1711</D>
<A>G. Cary</A> <W>Phys. Phyl.</W> 29 <T>My Salamandrous
Spirit..my &Ae.tnous burning Humours.</T></Q></QP></S6>
</S4><S4><DEF>So </DEF><SE><BL><LF>salamandry</LF><SF>
sala&sm.mandry</SF> <MF>salama&sd.ndry</MF></BL><DEF>
<PS>a.</PS></DEF></SE><QP><EQ><Q><D>1610</D> <A>Boys</A>
<W>Expos. Dom. Epist. &amp. Gosp.</W> Wks.  (1629) 76
<T>If a Salamandry spirit should traduce that godly labour,
as the silenced Ministers haue wronged our Communion Booke.
</T></Q></EQ></QP></S4></E>

<E><HG><HL><LF>understrife</LF><SF>&sm.understrife</SF>
<MF>u&sd.nderstrife </MF></HL><b>.</b> </HG><LB>poet.</LB>
<ET><XR><XL>under-</XL><HO>1</HO> <SN>5</SN> <SN>b</SN>.
</XR></ET> <S4><S6><DEF>Strife carried on upon the earth.
</DEF><QP><EQ><Q><D>C. 1611</D> <A>Chapman</A> <W>Iliad</W>
xx. 138 <T>We soon shall..send them to heaven, to settle
their abode With equals, flying under&dubh.strifes.</T>
</Q></EQ></QP></S6></S4></E>
```

Figure 2. Two marked up dictionary entries.

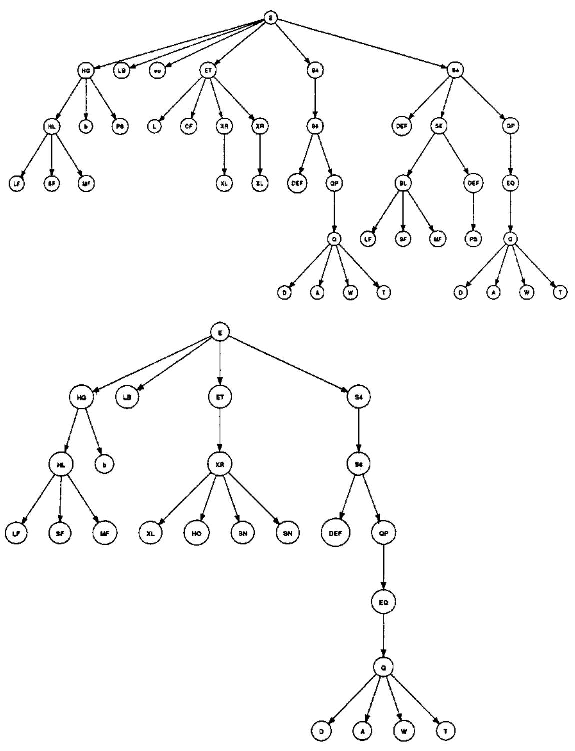


Figure 3. The two parse trees.

production occurs in the data is indicated by a preceding frequency. A non-terminal and all of its strings (right hand sides of its productions) can now be considered a training set for a single sub-problem. Note that if we generalize each such production to a regular expression then we have an overall grammar that is context free (Lalonde, 1977). This is the standard choice for modeling text structure.

Three existing grammatical inference approaches for text operate by specifying rules that are used to rewrite and combine strings in the training set (Chen, 1991; Fankhauser & Xu, 1993; Shafer, 1995). The following rule, for example, generalizes a grammar by expanding the language that it accepts:

- $ab^nc \rightarrow ab^+c$ if n is greater than or equal to a given value

Applied to the first entry in figure 2 with $n = 2$ this gives `E -> HG LB su ET S4+`, for example. Other rules have no effect on the language, but simplify a grammar by combining productions:

- $abc \wedge ac \rightarrow ab?c$
- $ab^+c \wedge ac \rightarrow ab^*c$.

```
1 : E -> HG LB su ET S4 S4
1 : E -> HG LB ET S4
1 : HG -> HL b PS
1 : HG -> HL b
1 : ET -> L CF XR XR
1 : ET -> XR
2 : S4 -> S6
1 : S4 -> DEF SE QP
2 : HL -> LF SF MF
2 : XR -> XL
1 : XR -> XL HO SN SN
2 : S6 -> DEF QP
1 : DEF -> PS
1 : SE -> BL DEF
1 : QP -> Q
2 : QP -> EQ
1 : BL -> LF SF MF
3 : Q -> D A W T
1 : EQ -> Q
```

Figure 4. The de facto grammar with production frequencies.

Applied to the two `HG` productions in figure 2, for example, the first rule gives `HG -> HL b PS?`.

Work by Ahonen (1996) and Ahonen, Mannila, and Nikunen (1994a, 1994b) uses a more classical grammatical inference approach of generating a language belonging to a characterizable subclass of the regular languages. Specifically, they use $(k-h)$ contextual languages, an extension that they propose to k-contextual languages.

In contrast to previous approaches to grammar construction for text structure, we use the frequency information from the training set to generate a stochastic grammar. The non-stochastic approaches mentioned above are inappropriate for larger document collections with complex structure. This is because there is no way to deal with the inevitable errors and pathological exceptions in such cases without considering frequency information. Stochastic grammars are also better suited for understanding and exploration since they express additional semantics and provide a natural way of distinguishing the more significant components of the grammar. The thesis by Ahonen (1996) does partially address such concerns. For example, when applying her method to a Finnish dictionary of similar complexity to the OED, she first removed all but the most frequent cases from the training set. She also proposes some ad-hoc methods for separating the final result into high and low frequency components (after having generated the result with a method that does not consider frequency information).

We assert that it is better to use an inference method that considers frequency information from the beginning. The stochastic grammatical inference algorithm that we have chosen to adopt for this application was proposed by Carrasco and Oncina (1994b). Our modifications are motivated by shortcomings in the ability to tune the algorithm and by our wish to use it interactively for exploration rather than as a black box to produce a single, final result. Note that understanding techniques are needed to support effective interaction. The primary technique used for examples in this paper is visualizing automata as bubble diagrams. All bubble diagrams were generated by the graph visualization program *da Vinci* which performs node placement and edge crossover minimization (Fröhlich & Werner, 1995).

The remainder of the paper is as follows: Section 2 describes the underlying algorithm, Section 3 explains our modifications, Section 4 evaluates the method, and Section 5 concludes.

2. Algorithm

Here we provide an overview of the algorithm by Carrasco and Oncina (1994b). Of the many possible stochastic grammatical inference algorithms, the particular one used here was chosen for several reasons. First of all, it is similar to the method of Ahonen et al. in that it uses a finite automaton state merging paradigm. Since that work represents the most in-depth examination of grammar inference for text structure to date, it is reasonable to use a similar approach. In fact, many of their results that go beyond just the application of the algorithm (such as rewriting the automaton into a grammar consisting of productions) can be adapted to the outputs of our algorithm. The second reason for choosing this algorithm was that the basic generalization operation of merging states is guided by a justifiable statistical test rather than an arbitrary heuristic. Note that the Bayesian model merging approach of Stolcke and Omohundro (1994) or a probability estimation approach based on the forward-backward algorithm (Huang, Ariki, & Jack, 1990; Lari & Young, 1990) are other candidates satisfying this particular characteristic.

2.1. ALERGIA

The algorithm ALERGIA by Carrasco and Oncina (1994b) is itself an adaptation of a non-stochastic method by Oncina and García (1992).

The algorithm produces stochastic finite automata (SFAs). These grammar constructs can be informally explained as finite automata that have probabilities associated with their transitions. The probability assigned to a string is, therefore, the product of the probabilities of all transitions followed in accepting it. Note that every state also has an associated termination probability, and that this is included in the product. Any state with a non-zero termination probability can be considered a final state. See the book by Fu (1982) for a more formal and complete description of SFAs and their properties.

The inference paradigm used by ALERGIA is a common one: first build a de facto model that accepts exactly the language of the training set; then generalize. Generalization for finite automata is done by merging states. This is similar to the state merging operation used in the algorithm for minimizing a non-stochastic finite automaton (Hopcroft & Ullman, 1979). The difference is that merges that change the accepted language are allowed.

Consider, as an example, the productions for ET from the training data of figure 4. These can be represented by the prefix tree in figure 5. The primitive operation of merging two

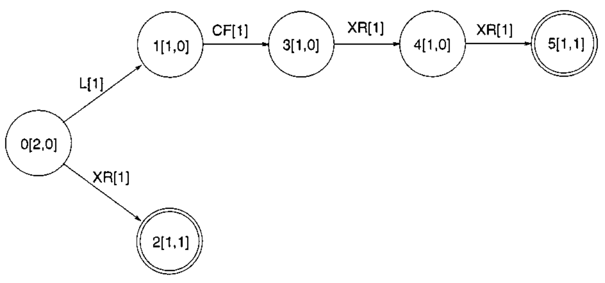


Figure 5. A de facto SFA (prefix tree) for the ET element. States are labeled ID[N,T] where ID is the state identifier, N is the incoming frequency, and T is the termination frequency. Transitions are labeled S[F], where S is the symbol, and F is the transition frequency. Final states with non-zero termination frequencies are marked with double rings.

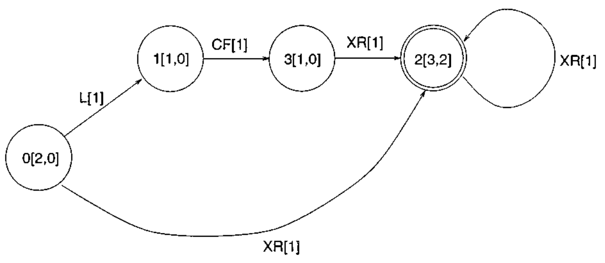


Figure 6. Figure 4 with states 2, 4 and 5 merged.

states replaces them with a single state, labeled by convention with the smaller of the two state identifiers. All incoming and outgoing transitions are combined and the frequencies associated with any transitions they have in common are added, as are the incoming and termination frequencies. Figure 6 demonstrates the effect of merging states 2 and 4, then 2 and 5.

Note that if two states have outgoing transitions with the same symbol but different destinations, these two destinations are also merged to avoid indeterminism. Thus, merging two non-leaf states can recursively require merging of long strings of their descendants.

An algorithm based on this paradigm must define two things: how to test whether two given states should be merged, and the order in which pairs of states are tested.

In ALERGIA, two states are merged if they satisfy the following equivalence critera: for every symbol in the alphabet, the associated transition *probabilities* from the states are equal; the termination *probabilities* for the states are equal; and the destination states of the two transitions for each symbol are equivalent according to a recursive application of the same criteria.

Whether two transitions' *probabilities* are equal is decided with a statistical test of the observed *frequencies*. Let the null hypothesis H_0 be that they are equal and the alternative H_a be that they differ. Let n_1, n_2 be the number of strings that arrive at the states and f_1, f_2 be the number of strings that follow the transitions in question (or terminate). Then, using the Hoeffding bound (Hoeffding, 1963) on binomial distributions, the p-value is less than a chosen significance level α if the test statistic

$$\left|\frac{f_1}{n_1} - \frac{f_2}{n_2}\right|$$

is greater than the expression

$$\sqrt{\frac{1}{2}\log\frac{2}{2\alpha}}\left(\frac{1}{\sqrt{n_1}} + \frac{1}{\sqrt{n_2}}\right).$$

In this case, reject the null hypothesis and assume the two probabilities are different, otherwise assume they are the same. This test ensures that the chosen α represents a bound on the probability of incorrectly rejecting the null hypothesis (i.e. incorrectly leaving two equivalent nodes separate). Thus, reducing α makes merges more likely and results in smaller models.

The order in which pairs of nodes are compared is defined as follows: nodes are numbered in a breadth-first order with all nodes at a given depth ordered lexically based on the string prefixes used to reach the node. Figure 5 is an example. Pairs of nodes (q_i, q_j) are tested by varying j from 1 to the number of nodes, and i from 0 to $j-1$. For the non-stochastic version of the algorithm, this ordering is necessary to prove identification in the limit (Oncina & García, 1992). Its significance in the stochastic version is unclear.

Note that the worst case time complexity of the algorithm is $O(n^3)$. This occurs for an input where no merges take place, thus requiring that all $n(n-1)/2$ pairs of nodes be compared, and furthermore, where the average recursive comparison continues to $O(n)$ depth. In practise, the expected running time is likely to be much less than this. Carrasco and Oncina report that, experimentally, the time increases linearly with the number of strings in the training set, as artificially generated by a fixed size model. We have observed a quadratic increase with the size of model. However, since this size is normally chosen, through parameter adjustment, to be small enough for understanding, running time has not been a problem in our experience.

3. Modifications

In this section we present our modifications to the algorithm, using the PQP (pseudo-quotation paragraph) element from the OED as an example. Of the 145,289 instances of that element in the entire dictionary, there are 90 unique arrangements for subelements; thus there are 90 unique strings that appear as right sides of productions in the de facto grammar. Those are shown in figure 7 along with their occurrence frequencies. The four elements that occur in a PQP are the EQ (earliest quotation), Q (quotation), SQ (subsidiary quotation), and LQ (latest quotation). The usage of the elements (which is known from the dictionary but can also be deduced from the examples) is as follows: SQ can occur any number of times and in any position; Q can occur any number of times; EQ can occur at most once and must occur before the first Q; LQ can occur at most once and must occur after the last

```
21:
524:    EQ
5:      EQ,LQ
294:    EQ,Q
1:      EQ,Q,LQ
156:    EQ,Q,Q
64:     EQ,Q,Q,Q
30:     EQ,Q,Q,Q,Q
1:      EQ,Q,Q,Q,Q,LQ
15:     EQ,Q,Q,Q,Q,Q
8:      EQ,Q,Q,Q,Q,Q,Q
5:      EQ,Q,Q,Q,Q,Q,Q,Q
3:      EQ,Q,Q,Q,Q,Q,Q,Q,Q
3:      EQ,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      EQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      EQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      EQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
2:      EQ,SQ
28:     LQ
80380:  Q
20:     Q,LQ
35824:  Q,Q
8:      Q,Q,LQ
15651:  Q,Q,Q
5:      Q,Q,Q,LQ
6335:   Q,Q,Q,Q
1:      Q,Q,Q,Q,LQ
2739:   Q,Q,Q,Q,Q
1166:   Q,Q,Q,Q,Q,Q
579:    Q,Q,Q,Q,Q,Q,Q
293:    Q,Q,Q,Q,Q,Q,Q,Q
124:    Q,Q,Q,Q,Q,Q,Q,Q,Q
93:     Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
46:     Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
28:     Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
10:     Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
9:      Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
4:      Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
4:      Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
2:      Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
2:      Q,Q,Q,Q,Q,SQ
1:      Q,Q,Q,Q,SQ,Q,Q,Q,Q,Q
4:      Q,Q,Q,SQ
2:      Q,Q,Q,SQ,Q,Q
1:      Q,Q,Q,SQ,Q,Q,Q
1:      Q,Q,Q,SQ,Q,Q,Q,Q,Q
1:      Q,Q,Q,SQ,Q,Q,Q,Q,Q,Q
17:     Q,Q,SQ
3:      Q,Q,SQ,Q
4:      Q,Q,SQ,Q,Q
3:      Q,Q,SQ,Q,Q,Q
2:      Q,Q,SQ,Q,Q,Q,Q
58:     Q,SQ
12:     Q,SQ,Q
1:      Q,SQ,Q,Q
6:      Q,SQ,Q,Q
9:      Q,SQ,Q,Q,Q
2:      Q,SQ,Q,Q,Q,Q
3:      Q,SQ,Q,Q,Q,Q,Q
2:      Q,SQ,Q,Q,Q,Q,Q,Q
1:      Q,SQ,Q,Q,Q,Q,Q,Q,Q
1:      Q,SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      Q,SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      Q,SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
2:      Q,SQ,SQ
22:     SQ
2:      SQ,EQ
5:      SQ,EQ,Q
5:      SQ,EQ,Q,Q
3:      SQ,EQ,Q,Q,Q
1:      SQ,EQ,Q,Q,Q,Q,Q
58:     SQ,Q
174:    SQ,Q,Q
77:    SQ,Q,Q,Q
102:    SQ,Q,Q,Q,Q
46:     SQ,Q,Q,Q,Q,Q
22:     SQ,Q,Q,Q,Q,Q,Q
9:      SQ,Q,Q,Q,Q,Q,Q,Q
8:      SQ,Q,Q,Q,Q,Q,Q,Q,Q
2:      SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q
2:      SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
2:      SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
1:      SQ,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q
2:      SQ,Q,Q,SQ
1:      SQ,Q,SQ,Q,Q,Q
```

Figure 7. The PQP example strings.

Q. This data is very simple and intended only to illustrate the modified learning algorithm. We give more complex examples in Section 4.2.

3.1. *Separation of low frequency components*

The original algorithm assumes that every node has a frequency high enough to be statistically compared. This is not typically valid. Nodes with too low a frequency always default to the null hypothesis of equivalence, resulting in inappropriate merges. The characteristic

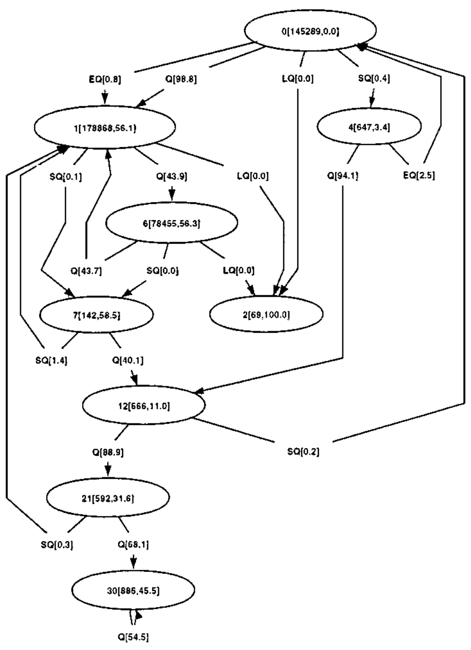


Figure 8. A result from the unmodified algorithm with transitions characteristic of inappropriate low frequency merges. Note that termination and transition frequencies f_i are shown converted to percentages representing f_i/n_i in this and subsequent figures to simplify comparisons.

result is that many low frequency nodes merge with either the root or another low index node (since the comparisons are made in order of index). This gives a structure with many inappropriate transitions pointing back to these low index nodes. Figure 8 shows an example result for the PQP data where transitions occur from several parts of the model to nodes 0 and 1.

Note that this form of inappropriate merging is not a problem that can be remedied just by tuning the single parameter α. As is usual in hypothesis tests, α is a bound on the probability of a false reject of the null hypothesis (i.e. failing to merge two nodes that are in fact equivalent). The complementary bound β on the probability of a false accept is unconstrained by the test of ALERGIA, and can be arbitrarily high for very low frequencies.

The problem can be seen as closely related to the small disjuncts problem discussed by Holte, Acker, and Porter (1989) for rule based classification algorithms: essentially, rules covering only a few cases of the training data perform relatively badly since they have inadequate statistical support. Holte et al. give three general approaches for improving the situation: 1) use exact significance tests in the learning algorithm, 2) test both significance and error rate of every disjunct in evaluating a result, and 3) whenever possible, use errors of omission instead of default classifications. Note that since our training set includes positive examples only, the second point does not apply. The first point, however corresponds to one of our modifications, and the third agrees with our own conclusions.

Experiments with several different treatments for low frequency nodes led us to the conclusion that no single approach would always produce an appropriate result (certainly not the original action of the algorithm—automatically merging on the first comparison).

This is understandable given that the frequencies in question are statistically insignificant. Therefore, we chose to first incorporate a significance test into the algorithm to separate out the low frequency nodes automatically, and then later decide on alternative treatments for these nodes (discussed in Section 3.4).

The following test is the standard one for checking equivalence of two binomial proportions while considering significance (see Desu (1990), for example). Assume as before that p_1, p_2 represent the unknown, true probabilities and that $f_1/n_1, f_2/n_2$ serve as the estimates. Sample sizes are required to satisfy the following relationship with α and β (which bound the probabilities of a false reject or a false accept of the null hypothesis):

$$\frac{1}{n_1} + \frac{1}{n_2} < \left\{ \frac{2\epsilon^*}{z_{\alpha/2} + z_\beta} \right\}^2$$

where z_x denotes the t value at which the cumulative probability of a standard normal distribution is equal to $1 - x$; and,

$$\epsilon^* = \arcsin\sqrt{0.50 + \epsilon/2} - \arcsin\sqrt{0.50 - \epsilon/2}.$$

The value ϵ is an additional parameter required to bound β (representing the minimum true difference between $p1$ and $p2$). The null hypothesis is rejected iff

$$z_{\alpha/2} > \frac{\left|2\arcsin\sqrt{f_1/n_1} - 2\arcsin\sqrt{f_2/n_2}\right|}{\sqrt{1/n_1 + 1/n_2}}.$$

We incorporate this test into the algorithm in the following way. Associate a boolean flag with each node, initially false; and, set the flag to true the first time a node is involved in a comparison with another node that satisfies the required relationship between α, β, and sample sizes. Nodes that still have false flags when the algorithm terminates are classified as low frequency components. An example result with the PQP data is shown in figure 9. Low frequency nodes in the graph are depicted as rectangles.

3.2. Control over the level of generalization

An important interactive operation is control over the level of generalization (how much the finite language represented by the training set is expanded). One possible approach is to vary α and β. Reducing α increases generalization: it restricts the possibility of incorrectly leaving nodes separate, and therefore makes merges more likely. Increasing β also increases generalization: it increases the allowable possibility of incorrectly merging nodes. Note that these two are not completely equivalent since α and β are *bounds* on the probabilities of their respective errors.

Unfortunately, it is not appropriate to control generalization in this way since α and β directly determine which components of the data are treated as too low frequency to be significant (i.e. the parts that will be merged by default using the original algorithm, or classified as low frequency according to the test in Section 3.1). Therefore, unless we are in a position to arbitrarily vary the amount of available data according to our choice of parameters, another modification is needed.

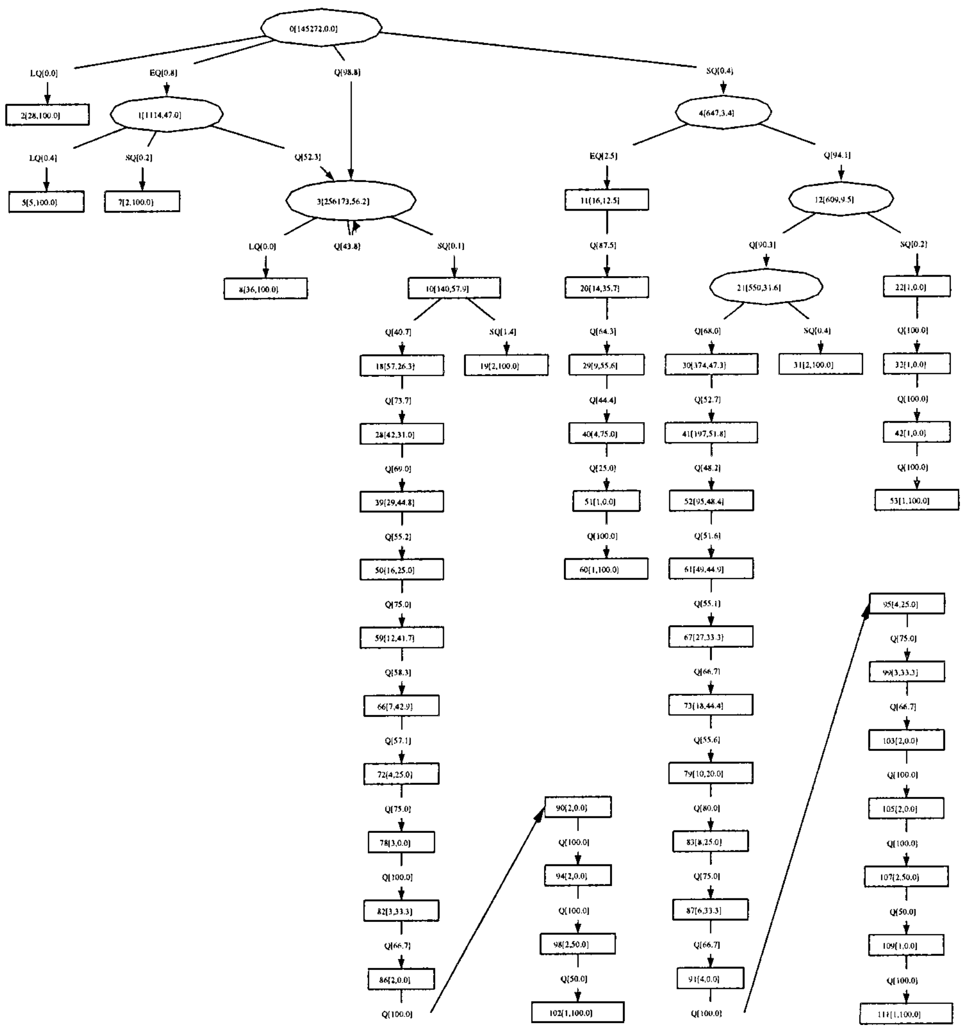


Figure 9. The PQP inference result with $\alpha = \beta = 0.025$.

The goal is to allow independent control over the division into low and high frequency components, and over the level of generalization. This is done by changing the hypothesis for the statistical test. Rather than testing whether two observed proportions can plausibly be equal, test whether they can plausibly differ by less than some parameter γ:

$$H_o : |p_1 - p_2| \leq \gamma$$
$$H_a : |p_1 - p_2| > \gamma$$

The modified test is as follows: let π_1 be f_1/n_1, and π_2 be f_2/n_2. Then, if

$$|\pi_1 - 0.5| < |\pi_2 - 0.5|$$

add $\gamma \cdot \mathrm{sign}(\pi_2 - \pi_1)$ to π_1, otherwise add $\gamma \cdot \mathrm{sign}(\pi_1 - \pi_2)$ to π_2. Reject H_0 iff

$$\frac{\left|2 \arcsin \sqrt{\pi_1} - 2 \arcsin \sqrt{\pi_2}\right|}{\sqrt{1/n_1 + 1/n_2}} > z_{\alpha/2}.$$

A larger value of γ results in a null hypothesis that is easier to satisfy, therefore producing more merges and an increase in generalization. As an example, consider the three results in figures 10–12 with constant α and β values, but varying γ values (and low frequency components clipped out for the moment). Higher γ values result in fewer nodes, larger languages, and less precise probability predictions.

3.3. *Choosing algorithm parameters*

The modified algorithm has the following parameters:

- γ is the maximum difference in true proportions for which the algorithm should merge two states.

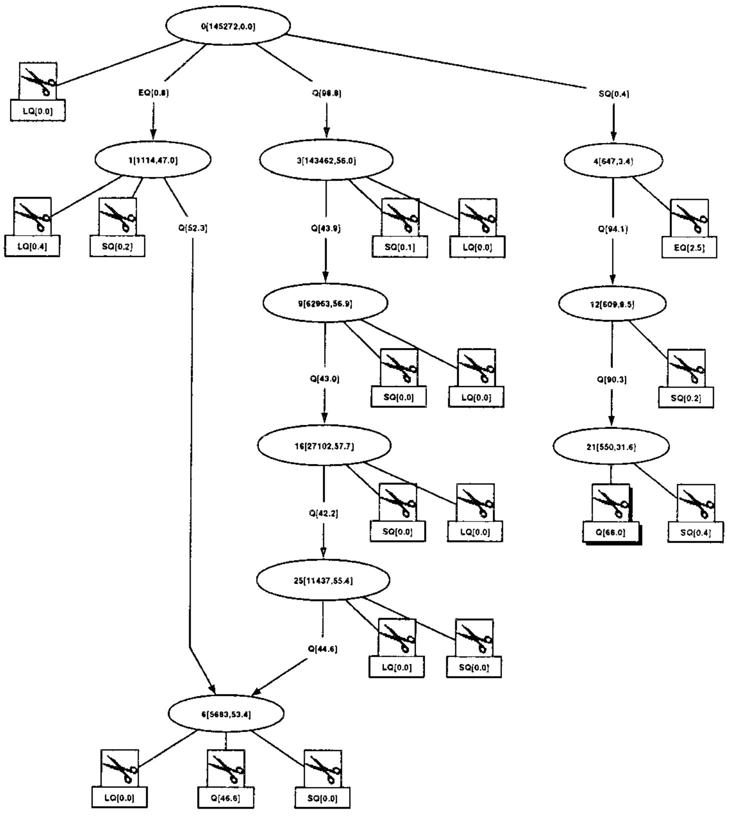


Figure 10. The PQP inference result with $\alpha = \beta = 0.025$, and $\gamma = 0.0$.

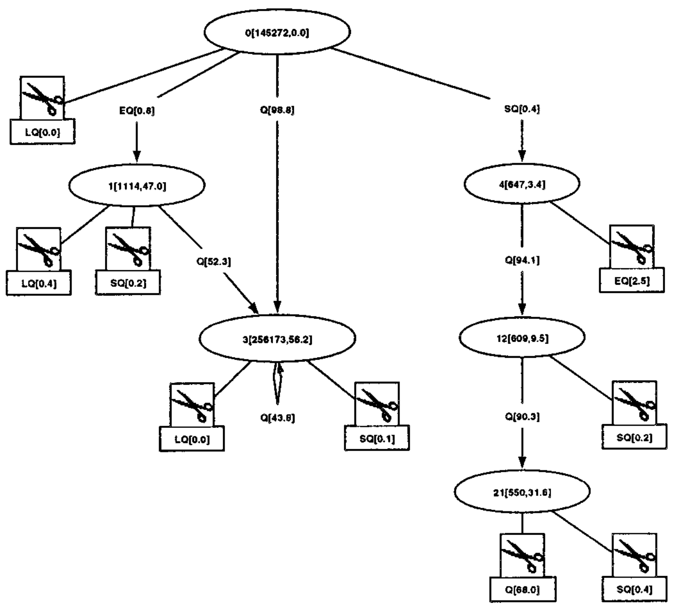


Figure 11. The PQP inference result with $\alpha = \beta = 0.025$, and $\gamma = 0.025$.

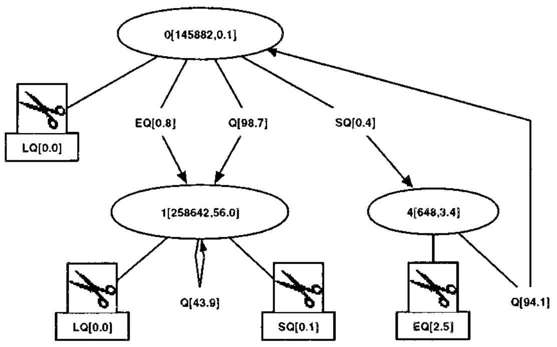


Figure 12. The PQP inference result with $\alpha = \beta = 0.025$, and $\gamma = 0.250$.

- α is the probability bound on the chance of making a type I error (incorrectly concluding that the two proportions differ by more than γ).
- β is the probability bound on the chance of making a type II error (incorrectly concluding that the two proportions differ by less than γ) when the true difference in the proportions is at least $\gamma + \epsilon$ (ϵ being the fourth parameter).

We next describe the effects of changing these parameters and also explain that useful interaction does not necessarily require separate control over all four.

Choosing γ controls the amount of generalization. Setting it to 0 results in very few states being merged; setting it to 1 always results in an output with a single state (effectively a 0-context average of the frequency of occurrence of every symbol).

Changes to α and β also affect the level of generalization. Their main effect of interest, however, is that they define the cutoff between high and low frequency components. Increasing either one decreases the number of nodes classified as low frequency. For simplified interaction, it is possible to always have both equal and adjust them together as a single parameter. This does not seriously reduce the useful level of control over the algorithm's behavior.

The ϵ parameter determines the difference to which the β probability applies. This must be specified somewhere but is not an especially useful value over which to have control. It should therefore be fixed, or tied in some way to the size of the input and the value of γ (we choose to fix it).

Overall then, it can be seen that control is only really needed over two major aspects of the inference process. Choosing a combined value for α and β sets the cutoff point between the significant data and the low frequency components. Choosing γ controls the amount of generalization.

As an example of parameter adjustment, consider the inference result from figure 9. Examination reveals two possible changes. The first is based on the observation that nodes 1 and 3 are very similar: they both accept an LQ or SQ or any number of Q's, and their transition and termination probabilities all differ by less than ten percent. Unless these slight differences are deemed significant enough to represent in the model, it is better to merge the two nodes. This can be done by increasing γ to 0.1, thus allowing nodes to test equal if their true probabilities differ by up to ten percent.

The second adjustment affects nodes 4, 12 and 21. These express the fact that strings starting with an SQ are much more likely to end with more than two Q's. This rule only applies, however, to about five hundred of the over one hundred forty five thousand PQPs in the dictionary. If we choose to simplify the model at the expense of a small amount of inaccuracy for these cases, we can reduce α and β to reclassify these nodes as low frequency. Bisection search of values of α and β between 0 and 0.025 reveals that this is accomplished with $\alpha = \beta = 0.005$. The result after application of the two adjustments described above is shown in figure 13.

3.4. What to do with the low frequency components

There are three possible ways of treating low frequency components: assume the most specific possible model by leaving the components separate (this is the same as leaving β fixed and allowing α to grow arbitrarily high); merge all the low frequency components into a single garbage state (an approach adopted in Ron, Singer, and Tishby (1995)); or, merge low frequency nodes into other parts of the automaton. Many methods can be invented for the last approach. We have observed that, in general, a single method does not produce appropriate results for all components of a given model. We therefore propose a *tentative merging* strategy. First an ordered list of heuristics is defined. Then all low frequency components are merged into positions in the model determined by the first heuristic in the list. If

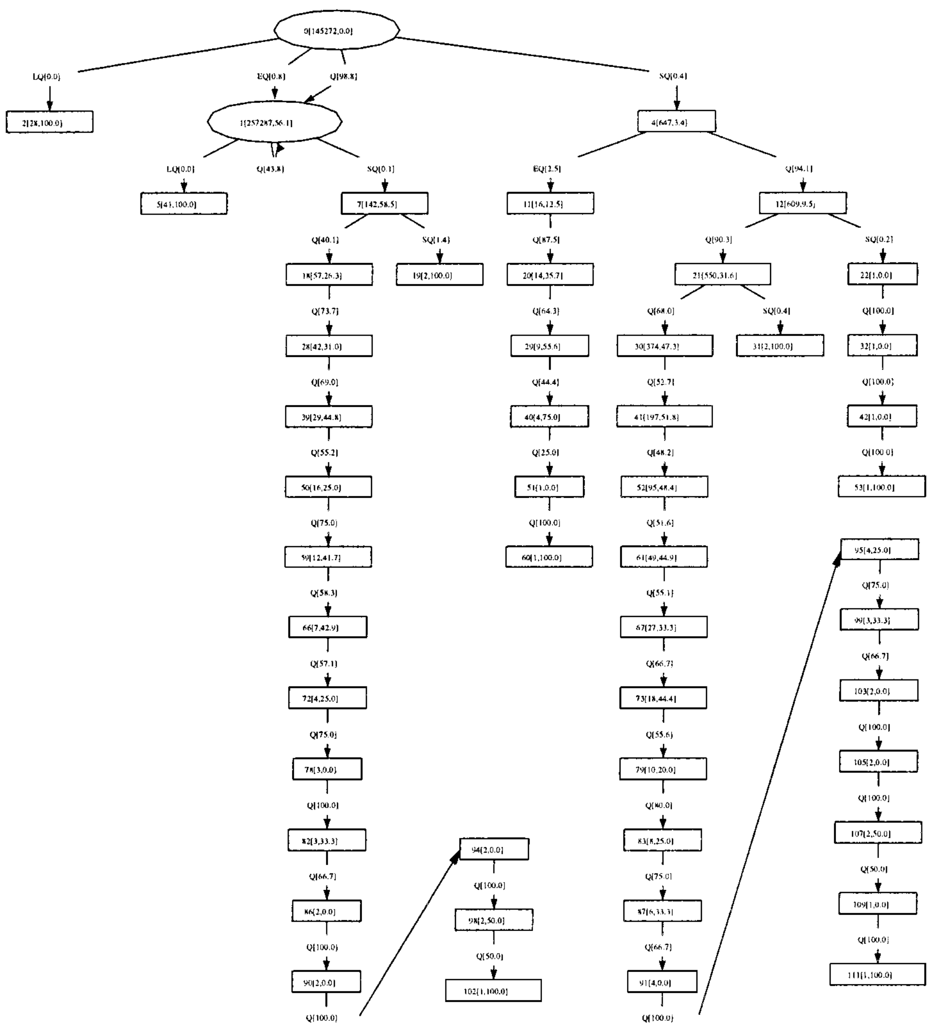


Figure 13. The PQP inference result with $\alpha = \beta = 0.005$ and $\gamma = 0.10$.

the user identifies a problem with a particular resulting *tentative transition* then the subtree can be re-merged into a position determined by the next heuristic in the list.

Heuristics can be designed based on various grammatical inference or learning approaches. Note that the problem of choosing a place to merge a low frequency component differs from the general problem of stochastic grammatical inference in two ways: 1) the rest of the high frequency model is available as a source of information, and 2) the frequency information has been classified as insignificant. The second point implies that, if we choose to consider frequency information, we may have to use special techniques to compensate. These could include a Laplace approximation of the probability or a Bayesian approach using a prior probability. Evidence measures developed for recent work in DFA (rather than SFA) learning might also be applicable (Lang, Pearlmutter, & Price, 1999).

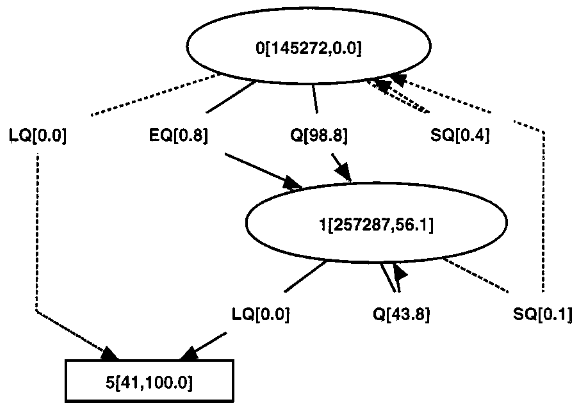


Figure 14. Figure 13 with low frequency components merged into other parts of the graph.

We mention two heuristics that do not use frequency information but that we have found to work well. Both guarantee that the model is still able to parse all strings in the training set. The first is to merge every low frequency node with its immediate parent. The result is that any terminals occurring in a low frequency subtree are allowed to occur any number of times and in any order starting at the nearest high-frequency ancestor. The second heuristic is to locate a position in the high frequency model from which an entire low-frequency subtree can be parsed. This subtree can then be merged into the rest of the model by replacing it with a single transition to the identified position. If more than one possible position exists, these can be stepped through before proceeding to the next heuristic in the list.

As an example of the application of the second heuristic reconsider figure 13. Merging every low frequency tree in that graph into the first (lowest index) node that can parse it gives the result in figure 14. Tentative transitions in that diagram are marked with dashed lines. The tentative transition from node 1 to 0 on input of SQ creates a cycle that allows more than one EQ to occur. This violates proper usage of that element as outlined in Section 3. Re-pointing the transition to node 1, an alternate destination that parses the low frequency subtree, gives an acceptable result for the PQP element.

4. Evaluation

In this section we compare the modified algorithm (henceforth referred to as MOD-ALERGIA) with ALERGIA. First, using data drawn from four different texts, we compare performance with automatic searches. Then we use two specific examples to illustrate some other points of comparison.

4.1. Batch experiments

We describe experiments demonstrating that, even with an automatic search procedure and simple default treatment of low frequency components, MOD-ALERGIA can be used to find probabilistically better models. The following four texts are used:

- OED—the *Oxford English Dictionary* (Simpson & Weiner, 1989). This is over 500 Mb and exhibits complex, sometimes irregular, structure.
- CPS—a pharmaceutical database which is an electronic version of a publication that describes all drugs available in Canada (Krogh, 1995). This is 18 Mb and exhibits a mix of simple and complex structure.
- OALD—the *Oxford Advanced Learner's Dictionary* (Hornby & Cowie, 1992). This is 17 Mb and exhibits complex structure that is more regular than the OED having been designed from the beginning for computerization.
- HOWTO—the SGML versions of HOWTO documents for the Linux operating system. This is 10 Mb and exhibits relatively simple structure.

It is not worth performing inference on terminal structural elements and non-terminals with very little sub-structure. As an arbitrary cutoff, we discard those that give de facto automata with fewer than 10 states. This leaves 24 elements in OED, 24 in CPS, 23 in OALD, and 14 in HOWTO for a total of 85 data sets. The procedure for each data set is as follows:

1. Randomly split the strings into equal size training, validation, and test sets.
2. Using the training set to generate the de facto automaton, and calculating goodness against the validation set (the metrics are detailed below), search the space of parameters in two ways:

   (A) Let x be the number of states of the de facto automaton for the training set. Using ALERGIA, test x different α values evenly spaced over the unit interval. (This is enough to find most of the possible models.)

   (B) Using MOD-ALERGIA and merging low frequency components with their immediate parents, test $x/2$ parameter values with $\beta = 1$, $\gamma = 0$, and α evenly spaced over the unit interval (which behaves the same as ALERGIA). Test the remaining (at most) $x/2$ parameter values with $\alpha = \beta$ and γ varied as follows:

```
max = $\lfloor\sqrt{x/2}\rfloor$
for $\gamma = 0.0$ to $1.0$ step $1.0/\max$
    for $\alpha = 0.0$ to $1.0$ step $1.0/\max$
        run MOD-ALERGIA
```

3. Evaluate the best model found by each method using the test set.

We perform the above procedure using two different metrics. The first is found by totaling the probability of all strings that are assigned zero probability by the model. We call this the error since it corresponds to the probability of rejecting a valid string if the SFA is stripped of its probabilities and used as a DFA.

The second metric is cross entropy (also called *Kullback-Liebler divergence*) which quantifies the probabilistic fit of one model to another. This measure has previously been used to evaluate stochastic grammatical inference methods (Sánchez & Benedí, 1994). Given two probabilistic language models, M_1 and M_2, the cross entropy is defined as:

$$H(M_1, M_2) = \sum_{\forall x \in L} P_{M_1}(x) \cdot \log \frac{P_{M_1}(x)}{P_{M_2}(x)}.$$

where $P_{M_1}(x)$ is the probability in the set and $P_{M_2}(x)$ is the probability assigned by the model. We calculate this with L first equal to the validation set and later the test set. To avoid assigning zero probabilities to strings not recognized by the model, we smooth the distribution in the following way:

- Add a "dead" state that has transitions returning to itself for all symbols in the alphabet. Give all these transitions equal probability.
- For each state, add an outgoing transition to the dead state for every symbol in the alphabet not already present on an outgoing transition. Set the probability of these transitions to a small constant p_{dead}.
- For each state, if the termination probability is 0, change it to p_{dead}.
- Normalize every state so that the sum of the termination probability and all outgoing probabilities is 1.0.

A single, constant value for p_{dead} was used for all 85 data sets. It was chosen empirically so that all absent strings (i.e. strings that were zero probability before smoothing) were assigned smaller probabilities after smoothing than all non-absent strings.

On a SUN supersparc, the runs averaged 11 minutes for each of the 85 elements. The average error for ALERGIA over all 85 elements was 0.028; for MOD-ALERGIA it was 0.024. When considering elements individually, the average improvement was 64.6 percent. A t test that the mean improvement was greater than 0 gave a p-value of 8e-17. Similarly, using cross entropy, the average value for ALERGIA was 7.1, the average for MOD-ALERGIA was 5.1, the average individual improvement was 18.1 percent, and the p-value was 8e-8.

There are some characterizable performance differences between the 85 data sets. For example, if we sort them according to the average node frequency in the de facto automaton (which is the same as the sum of all string lengths in the training set divided by the number of states in the de facto automaton), and then break them into three equal size sets, we get the result in Table 1. This shows that the modified algorithm does relatively better for data sets with lower frequencies. This is partly because the performance of ALERGIA is worse in these cases, and partly because the modifications in MOD-ALERGIA are targeted to deal with low frequency cases.

Overall, these experiments clearly show that the extra parameter dimensions available to MOD-ALERGIA provide the ability to find better models with realistic computing effort. The given simple search procedure does significantly better on average, and in fact, the only cases where the MOD-ALERGIA search did worse were ones where it had obviously over-fitted the validation set. With a search procedure designed to avoid over-fitting, MOD-ALERGIA could be used exclusively on a new data set with the expectation of always doing as well or better than ALERGIA.

Table 1. Error and cross entropy comparison for different average node frequencies. Part (a) is error; part (b) is cross entropy. The first column gives the range of average node frequencies. The second column is the average metric value over all 85 elements for ALERGIA; the third column for MOD-ALERGIA. Column four is the mean individual improvement. Column five is the p-value for a t test that the mean improvement is greater than 0.

Avg. node freq.	ALERGIA	MOD-ALERGIA	% improve	(p-value)
(a) Average error				
0–19	0.031	0.024	72.4	(3e-9)
19–101	0.035	0.032	62.8	(1e-7)
101+	0.018	0.018	59.0	(8e-7)
(b) Average cross entropy				
0–19	12.9	7.25	24.1	(1e-2)
19–101	5.07	4.90	16.5	(6e-5)
101+	3.07	3.06	14.2	(7e-4)

4.2. Particular examples

This section gives two examples that demonstrate some other advantages of the stochastic inference approach in general, and of the modified algorithm in particular.

For the first example we use the Entry element from the OED and create an over-generalized model to compare with the prototypical entry presented in figure 1. ALERGIA cannot produce any models for this element in the size range of 2 to 13 nodes (even using a bisection search for values of α that narrows the search interval all the way down to adjacent numbers in a double precision floating point representation). In contrast, the MOD-ALERGIA search described in the previous section generates a model for every size in this interval. (Note that the MOD-ALERGIA search also generated about 70 percent more different sized models overall.) After inspection of a few of the smaller models for Entry, we found the seven node graph shown in figure 15 to be most similar to the prototypical entry. This model highlights several interesting characteristics. One of the paths (HG VL ET S4*) does correspond to the prototypical entry, but note some of the semantics that are not present in the non-stochastic description:

- The variant form list (VL) is optional and is actually omitted more often than not.
- The etymology (ET) can also be omitted, skipping directly to the senses. Most often this does not happen.
- An element not mentioned in figure 1, the status (ST), frequently precedes the headword group (HG) and its presence significantly increases the chance that the ET and VL will be bypassed. If they are not bypassed, however, a label (LB) element is normally inserted between them and the HG.
- Any number of LBs can also occur in an entry without an ST. Usually, however, not many occur (the loop probability is only 0.262).

Properties such as these can be extremely useful when it comes to exploring and understanding the data, even if they are disregarded for more standard grammar applications such

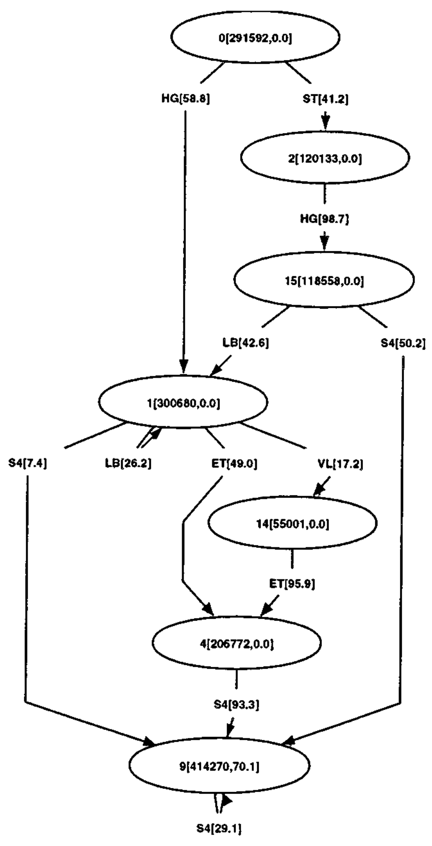


Figure 15. An inference result for the Entry element from the OED.

as validating a document instance. Furthermore, the stochastic properties of the grammar can be used to exercise editorial control when new entries are introduced into the dictionary: patterns that rarely occur can first trigger a message to the operator to double-check for correctness; if asserted to be what was intended they can be entered, yet flagged for subsequent review and approval by higher-level editorial authorities.

In our second example we use the Monograph element from the CPS data to again comment on the advantage of separating low frequency components (we have already done this for the PQP example). Figure 16 shows the first three high frequency nodes of a model for this data. Outgoing low frequency components are shown clipped. To get a final detailed model, we need to expand and examine the subtrees of these low frequency components one at a time. For each subtree we have the option of interactively stepping through positions where it can merge (for instance the immediate parent, and all other nodes from which it can be parsed), deciding to change the inference parameters to reclassify part of it as high frequency, or deciding that it represents an error in the data. This type of interactive correction is not possible with unmodified ALERGIA.

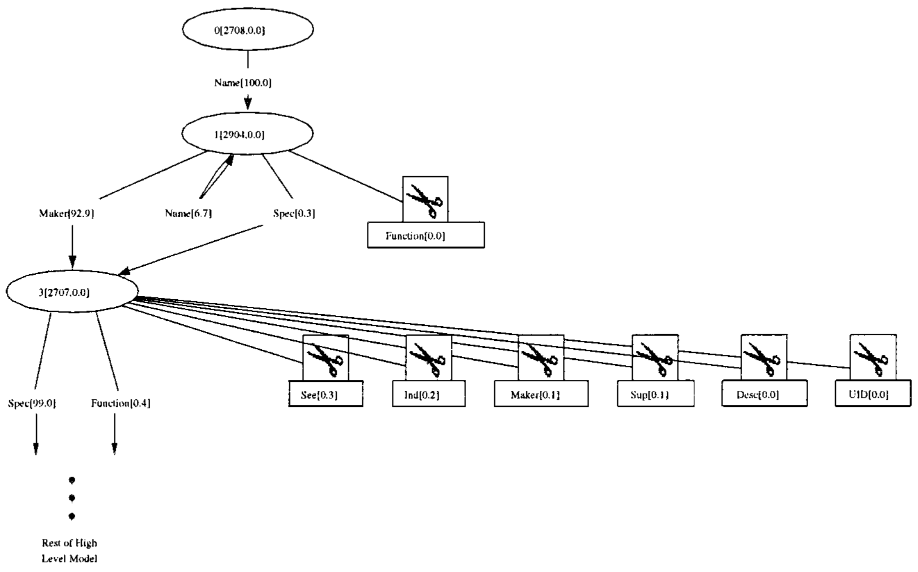


Figure 16. The first three nodes of the Monograph element from the CPS data. All clipped components indicated with scissors are low frequency components.

5. Conclusions and future work

This study was concerned with the application of grammatical inference to text structure, a subject that has been addressed before (Ahonen, 1996; Ahonen et al., 1994a,1994b; Chen, 1991; Fankhauser & Xu, 1993; Shafer, 1995). Grammar generation can be an important tool for maintaining a document database system. It is useful for creating grammars for standard text database purposes, but also allows a more flexible view. Rather than having a fixed grammar that describes all possible forms of the data, the grammar is fluid and evolves. Not only does the grammar change as new data is added, but many different forms of the grammar can be generated at any time, an over-generalized high-level view or a description for a subset of the data, for example. Thus we can generate grammars as much to summarize and understand the organization of the text as to serve in traditional capacities like a schema.

Our approach adds two things to previous approaches: extension to stochastic grammatical inference, and an algorithm with greater freedom for interactive tuning. The advantages of changing to stochastic inference are as follows:

- Stochastic inference is more effective since it uses frequency information as part of the inference process. This is true for any learning method.
- Stochastic models have richer semantics and are therefore easier to interpret and interactively adjust. This was demonstrated with the Entry example in Section 4.2. Note that stochastic models can easily be converted to non-stochastic ones by dropping the probability information. Therefore, we are free to use the algorithm just as a more effective method for learning non-stochastic models.

- A stochastic inference framework allows parameterization that can be used to produce different models for a single data set. This flexibility can be used to search for a single best model, or to explore several models at different generalization levels. Existing non-stochastic approaches to this problem all work as black boxes producing a single, untunable result.

The additional tunability of the modified algorithm was shown to be useful in two ways: an experimental evaluation using four different texts, and two examples using specific elements from those texts.

Possibilities exist for further improvement of the algorithm. For example, the state merging paradigm for learning finite automata has seen some development since ALERGIA was first published. In particular, a control strategy that compares and merges nodes in a non-fixed order has been developed (Lang, Pearlmutter, & Price, 1999). This gives more freedom to merge nodes in order of the evidence supporting the merges. Incorporating it into our algorithm would be straightforward, especially in view of the fact that it is trivial to convert the result of a statistical test to an evidence measure. Another improvement would be to develop evidence measures to assist the user in choosing merge destinations for low frequency components. Possible starting points were mentioned in Section 3.4.

Much future work exists integrating the method into a system to support traditional applications. For example, the semi-structured database system Lore (McHugh et al., 1997) does generate schemas for use in query planning and optimization but performs no generalization, effectively stopping at the prefix tree. The schemas are therefore not necessarily compact nor understandable.

In addition to traditional applications, the stochastic part of the grammar also suggests many novel applications. For example, a system could be constructed to assist authors in the creation of documents by flagging excessively rare structures in the process of their creation, or listing possible next elements of partially complete entries in order of their probability. Stochastic grammars could also be used as structural classifiers by characterizing the authoring styles of two or more people who use the same tag set.

Acknowledgments

Financial assistance from the Natural Sciences and Engineering Research Council of Canada through a postgraduate scholarship, the Information Technology Research Center (now, Communications and Information Technology Ontario), and the University of Waterloo is gratefully acknowledged. We would also like to thank the anonymous reviewers for their detailed advice on improving the experimental evaluation.

References

Ahonen, H. (1996). Generating grammars for structured documents using grammatical inference methods. Ph.D. Thesis, University of Helsinki, Department of Computer Science, Technical Report A-1996-4.

Ahonen, H., Mannila, H., & Nikunen, E. (1994a). Forming grammars for structured documents: An application of grammatical inference. In Carrasco and Oncina (1994a), pp. 153–167.

Ahonen, H., Mannila, H., & Nikunen, E. (1994b). Generating grammars for SGML tagged texts lacking DTD. In M. Murata & H. Gallaire (Eds.), *Proc. Workshop on Principles of Document Processing (PODP), Darmstadt*.

André, J., Furuta, R., & Quint, V. (1989). *Structured documents*. The Cambridge Series on Electronic Publishing. Cambridge University Press.

Berg, D. L. (1989). The research potential of the electronic OED2 database at the University of Waterloo: a guide for scholars. Technical Report OED-89-02, UW Centre for the New Oxford English Dictionary, Waterloo, Ontario. available at `http://www.chass.utoronto.ca:8080/cch/Berg/Berg-1_Contents.html`.

Berg, D. L. (1993). *A guide to the Oxford English dictionary: The essential companion and user's guide*, Oxford University Press, Oxford.

Bray, T., Paoli, J., & Sperberg-McQueen, C. M. (1998). Extensible markup language (XML) 1.0. w3c recommendation.

Carrasco, R. & Oncina, J. (Eds.). (1994a). Lecture Notes in Computer Science, Vol. 862, Springer-Verlag.

Carrasco, R. C. & Oncina, J. (1994b). Learning stochastic regular grammars by means of a state merging method. In Carrasco and Oncina (1994a), pp. 139–152.

Chen, J. (1991). Grammar generation and query processing for text databases. Research proposal, University of Waterloo.

Clark, D. (1991). Finite state transduction tools. Technical Report OED-91-03, UW Centre for the New Oxford English Dictionary and Text Research, Waterloo, Ontario.

Coombs, J. H., Renear, A. H., & DeRose, S. J. (1987). Markup systems and the future of scholarly text processing. *Communications of the ACM, 30*(11), 933–947.

Desu, M. (1990). *Sample size methodology*. Boston: Academic Press.

Fankhauser, P. & Xu, Y. (1993). *MarkItUp!* an incremental approach to document structure recognition. *Electronic Publishing—Origin, Dissemination and Design, 6*(4), 447–456.

Fröhlich, M. & Werner, M. (1995). *da Vinci 1.4 User Manual*. available from `daVinci@informatik.uni-bremen.de`.

Fu, K. S. (1982). *Syntactic pattern recognition and applications*. Englewood Cliffs, N.J: Prentice-Hall.

Goldman, R. & Widom, J. (1997). DataGuides: Enabling query formulation and optimization in semistructured databases. In *Proceedings of the Twenty-Third International Conference on Very Large Databases* (pp. 436–445). Athens, Greece.

Gonnet, G. & Tompa, F. W. (1987). Mind your grammar: a new approach to modelling text. *Very Large Data Bases (VLDB), 13*, 339–346.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *American Statistical Association Journal, 58*, 13–30.

Holte, R. C., Acker, L. E., & Porter, B. W. (1989). Concept learning and the problem of small disjuncts, Sridharan, N. (Ed.), *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence V. 1*, Detroit, Michigan.

Hopcroft, J. E. & Ullman, J. D. (1979). *Introduction to automata theory, languages and computation*. Reading, Massachusetts: Addison-Wesley.

Hornby, A. S. & Cowie, A. P. (1992). *Oxford advanced learner's dictionary of current English 4th edn.* Oxford: Oxford University Press.

Huang, X., Ariki, Y., & Jack, M. (1990). *Hidden markov models for speech recognition*. Edinburgh University Press.

ISO (1986). Information processing—text and office systems—standard generalized markup language (SGML).

Kazman, R. (1986). Structuring the text of the *Oxford English dictionary* through finite state transduction. Technical Report CS-86-20, University of Waterloo, Computer Science Department.

Kilpeläinen, P., Lindén, G., Mannila, H., & Nikunen, E. (1990). A structured document database system. In Furuta, R. (Ed.), In *EP90—Proceedings of the International Conference on Electronic Publishing, Document Manipulation & Typography*, The Cambridge Series on Electronic Publishing. Cambridge University Press.

Krogh, C. M. (Ed.). (1995). *Compendium of pharmaceuticals and specialties, 30th edn.* Canadian Pharmaceutical Association.

Lalonde, W. (1977). Regular right part grammars and their parsers. *Communications of the ACM, 20*(10), 731–741.

Lang, K. J., Pearlmutter, B. A., & Price, R. A. (2000). Results of the Abbadingo one DFA learning competition and a new evidence driven state merging algorithm To appear.

Lari, K. & Young, S. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Comp. Speech and Language, 4*, 34–36.

McHugh, J., Abiteboul, S., Goldman, R., Quass, D., & Widom, J. (1997). Lore: a database management system for semistructured data. *SIGMOD Record, 26*(3), 54–66.

Muslea, I., Minton, S., & Knoblock, C. (1998). Wrapper induction for semistructured, web-based information sources. In *Proceedings of the Conference on Automatic Learning and Discovery (CONALD-98)*.

Oncina, J. & García, P. (1992). Inferring regular languages in polynomial updated time. In N. P. de la Blanca, A. Sanfeliu, & E. Vidal (Eds.), *Pattern Recognition and Image Analysis*, pp. 49–61. World Scientific.

Oxford University Press (1998). Inside the *Oxford English Dictionary*. `http://www.oed.com`.

Raymond, D. R. (1993). Visualizing text. In *9th Annual Conference of the UW Centre for the New OED*, Oxford, England. available at `http://solo.uwaterloo.ca/~drraymon/papers/oed93.ps`.

Ron, D., Singer, Y., & Tishby, N. (1995). On the learnability and usage of acyclic probabilistic automata. In *Computational Learning Theory, COLT 95* (pp. 31–40).

Sánchez, J. A. & Benedí, J. M. (1994). Statistical inductive learning of regular formal languages. In Carrasco and Oncina (1994a), pp. 130–138.

Shafer, K. (1995). Creating DTDs via the GB-engine and Fred. OCLC Fred web page `http://www.oclc.org/fred`.

Simpson, J. & Weiner, E. (Eds.). (1989). *The Oxford English dictionary*, 2nd edn. Oxford: Clarendon Press.

Stolcke, A. & Omohundro, S. (1994). Inducing probabilistic grammars by Bayesian model merging. In Carrasco and Oncina (1994a), pp. 106–118.

Suciu, D. (Ed.). (1997). In *Proc. of the Workshop on Managment of Semi-structured Data (PODS/SIGMOD)*, Tucson, Arizona. National Science Foundation. available at `http://www.research.att.com/~suciu/workshop-papers.html`.

Vidal, E. (1994). Grammatical inference: an introductory survey. In Carrasco and Oncina (1994a), pp. 1–4.

Received January 23, 1998
Accepted June 11, 1999
Final manuscript June 11, 1999