Bayesian Grammar Induction for Language Modeling

Stanley F. Chen
Aiken Computation Laboratory
Division of Applied Sciences
Harvard University
Cambridge, MA 02138
sfc@das.harvard.edu

Abstract

We describe a corpus-based induction algorithm for probabilistic context-free grammars. The algorithm employs a greedy heuristic search within a Bayesian framework, and a post-pass using the Inside-Outside algorithm. We compare the performance of our algorithm to n-gram models and the Inside-Outside algorithm in three language modeling tasks. In two of the tasks, the training data is generated by a probabilistic context-free grammar and in both tasks our algorithm outperforms the other techniques. The third task involves naturally-occurring data, and in this task our algorithm does not perform as well as n-gram models but vastly outperforms the Inside-Outside algorithm.

1 Introduction

In applications such as speech recognition, handwriting recognition, and spelling correction, performance is limited by the quality of the language model utilized (Bahl et al., 1978; Baker, 1975; Kernighan et al., 1990; Srihari and Baltus, 1992). However, static language modeling performance has remained basically unchanged since the advent of n-gram language models forty years ago (Shannon, 1951). Yet, n-gram language models can only capture dependencies within an n-word window, where currently the largest practical n for natural language is three, and many dependencies in natural language occur beyond a three-word window. In addition, n-gram models are extremely large, thus making them difficult to implement efficiently in memory-constrained applications.

An appealing alternative is grammar-based language models. Language models expressed as a probabilistic grammar tend to be more compact than n-gram language models, and have the ability to model long-distance dependencies (Lari and Young, 1990; Resnik, 1992; Schabes, 1992). However, to date there has been little success in constructing grammar-based language models competitive with n-gram models in problems of any magnitude.

In this paper, we describe a corpus-based induction algorithm for probabilistic context-free grammars that outperforms n-gram models and the Inside-Outside algorithm (Baker, 1979) in medium-sized domains. This result marks the first time a grammar-based language model has surpassed n-gram modeling in a task of at least moderate size. The algorithm employs a greedy heuristic search within a Bayesian framework, and a post-pass using the Inside-Outside algorithm.

2 Grammar Induction as Search

Grammar induction can be framed as a search problem, and has been framed as such almost without exception in past research (Angluin and Smith, 1983). The search space is taken to be some class of grammars; for example, in our work we search within the space of probabilistic context-free grammars. The objective function is taken to be some measure dependent on the training data; one generally wants to find a grammar that in some sense accurately models the training data.

Most work in language modeling, including n-gram models and the Inside-Outside algorithm, falls under the maximum-likelihood paradigm, where one takes the objective function to be the likelihood of the training data given the grammar. However, the optimal grammar under this objective function is one which generates only strings in the training data and no other strings. Such grammars are poor language models, as they *overfit* the training data and do not model the language at large. In n-gram models and the Inside-Outside algorithm, this issue is evaded by bounding the size and form of the grammars considered, so that the "optimal" grammar cannot be expressed. However, in our work we do not wish to limit the size of the grammars considered.

The basic shortcoming of the maximum-likelihood objective function is that it does not encompass the

$$
\begin{array}{llll}
S & \rightarrow & SX & (1-\epsilon) \\
S & \rightarrow & X & (\epsilon) \\
X & \rightarrow & A & (p(A)) \quad \forall\, A \in N - \{S, X\} \\
A_a & \rightarrow & a & (1) \quad \forall\, a \in T
\end{array}
$$

N = the set of all nonterminal symbols
T = the set of all terminal symbols

Probabilities for each rule are in parentheses.

Table 1: Initial hypothesis grammar

compelling intuition behind Occam's Razor, that simpler (or smaller) grammars are preferable over complex (or larger) grammars. A factor in the objective function that favors smaller grammars over large can prevent the objective function from preferring grammars that overfit the training data. Solomonoff (1964) presents a Bayesian grammar induction framework that includes such a factor in a motivated manner.

The goal of grammar induction is taken to be finding the grammar with the largest *a posteriori* probability given the training data, that is, finding the grammar G' where

$$G' = \arg\max_G p(G|O)$$

and where we denote the training data as O, for *observations*. As it is unclear how to estimate $p(G|O)$ directly, we apply Bayes' Rule and get

$$G' = \arg\max_G \frac{p(O|G)p(G)}{p(O)} = \arg\max_G p(O|G)p(G)$$

Hence, we can frame the search for G' as a search with the objective function $p(O|G)p(G)$, the likelihood of the training data multiplied by the prior probability of the grammar.

We satisfy the goal of favoring smaller grammars by choosing a prior that assigns higher probabilities to such grammars. In particular, Solomonoff proposes the use of the *universal a priori probability* (Solomonoff, 1960), which is closely related to the minimum description length principle later proposed by (Rissanen, 1978). In the case of grammatical language modeling, this corresponds to taking

$$p(G) = 2^{-l(G)}$$

where $l(G)$ is the length of the description of the grammar in bits. The universal *a priori* probability has many elegant properties, the most salient of which is that it dominates all other enumerable probability distributions multiplicatively.[1]

[1] A very thorough discussion of the universal *a priori* probability is given by Li and Vitányi (1993).

3 Search Algorithm

As described above, we take grammar induction to be the search for the grammar G' that optimizes the objective function $p(O|G)p(G)$. While this framework does not restrict us to a particular grammar formalism, in our work we consider only probabilistic context-free grammars.

We assume a simple greedy search strategy. We maintain a single hypothesis grammar which is initialized to a small, trivial grammar. We then try to find a modification to the hypothesis grammar, such as the addition of a grammar rule, that results in a grammar with a higher score on the objective function. When we find a superior grammar, we make this the new hypothesis grammar. We repeat this process until we can no longer find a modification that improves the current hypothesis grammar.

For our initial grammar, we choose a grammar that can generate any string, to assure that the grammar can cover the training data. The initial grammar is listed in Table 1. The sentential symbol S expands to a sequence of X's, where X expands to every other nonterminal symbol in the grammar. Initially, the set of nonterminal symbols consists of a different nonterminal symbol expanding to each terminal symbol.

Notice that this grammar models a sentence as a sequence of independently generated nonterminal symbols. We maintain this property throughout the search process, that is, for every symbol A' that we add to the grammar, we also add a rule $X \rightarrow A'$. This assures that the sentential symbol can expand to every symbol; otherwise, adding a symbol will not affect the probabilities that the grammar assigns to strings.

We use the term *move set* to describe the set of modifications we consider to the current hypothesis grammar to hopefully produce a superior grammar. Our move set includes the following moves:

Move 1: Create a rule of the form $A \rightarrow BC$

Move 2: Create a rule of the form $A \rightarrow B|C$

For any context-free grammar, it is possible to express a weakly equivalent grammar using only rules of these forms. As mentioned before, with each new

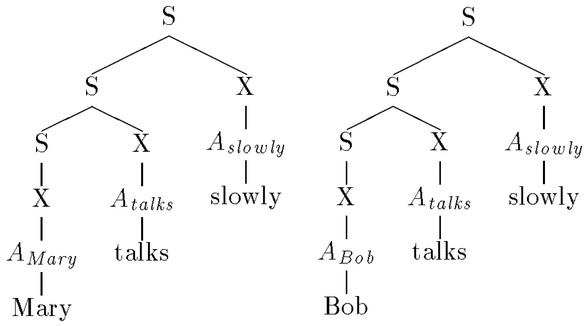

Figure 1: Initial Viterbi Parse

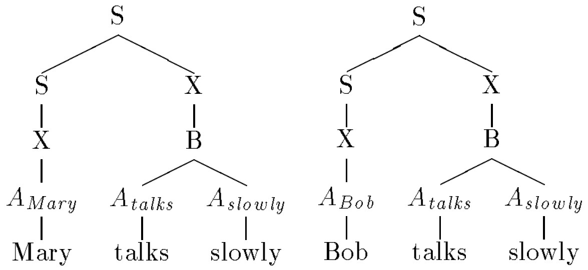

Figure 2: Predicted Viterbi Parse

symbol A we also create a rule $X \rightarrow A$.

3.1 Evaluating the Objective Function

Consider the task of calculating the objective function $p(O|G)p(G)$ for some grammar G. Calculating $p(G) = 2^{-l(G)}$ is inexpensive[2]; however, calculating $p(O|G)$ requires a parsing of the entire training data. We cannot afford to parse the training data for each grammar considered; indeed, to ever be practical for data sets of millions of words, it seems likely that we can only afford to parse the data once.

To achieve this goal, we employ several approximations. First, notice that we do not ever need to calculate the actual value of the objective function; we need only to be able to distinguish when a move applied to the current hypothesis grammar produces a grammar that has a higher score on the objective function, that is, we need only to be able to calculate the *difference* in the objective function resulting from a move. This can be done efficiently if we can quickly approximate how the probability of the training data changes when a move is applied.

To make this possible, we approximate the probability of the training data $p(O|G)$ by the probability of the single most probable parse, or *Viterbi* parse, of the training data. Furthermore, instead of recalculating the Viterbi parse of the training data from scratch when a move is applied, we use heuristics to predict how a move will change the Viterbi parse. For example, consider the case where the training data consists of the two sentences

$$O = \{\text{Bob talks slowly}, \text{Mary talks slowly}\}$$

In Figure 1, we display the Viterbi parse of this data under the initial hypothesis grammar used in our algorithm.

Now, let us consider the move of adding the rule

$$B \rightarrow A_{talks}\, A_{slowly}$$

to the initial grammar (as well as the concomitant rule $X \rightarrow B$). A reasonable heuristic for predicting how the Viterbi parse will change is to replace adjacent X's that expand to A_{talks} and A_{slowly} respectively with a single X that expands to B, as displayed in Figure 2. This is the actual heuristic we use for moves of the form $A \rightarrow BC$, and we have analogous heuristics for each move in our move set. By predicting the differences in the Viterbi parse resulting from a move, we can quickly estimate the change in the probability of the training data.

Notice that our predicted Viterbi parse can stray a great deal from the actual Viterbi parse, as errors can accumulate as move after move is applied. To minimize these effects, we process the training data incrementally. Using our initial hypothesis grammar, we parse the first sentence of the training data and search for the optimal grammar over just that one sentence using the described search framework.

[2]Due to space limitations, we do not specify our method for encoding grammars, *i.e.*, how we calculate $l(G)$ for a given G. However, this will be described in the author's forthcoming Ph.D. dissertation.

We use the resulting grammar to parse the second sentence, and then search for the optimal grammar over the first two sentences using the last grammar as the starting point. We repeat this process, parsing the next sentence using the best grammar found on the previous sentences and then searching for the best grammar taking into account this new sentence, until the entire training corpus is covered.

Delaying the parsing of a sentence until all of the previous sentences are processed should yield more accurate Viterbi parses during the search process than if we simply parse the whole corpus with the initial hypothesis grammar. In addition, we still achieve the goal of parsing each sentence but once.

3.2 Parameter Training

In this section, we describe how the parameters of our grammar, the probabilities associated with each grammar rule, are set. Ideally, in evaluating the objective function for a particular grammar we should use its optimal parameter settings given the training data, as this is the full score that the given grammar can achieve. However, searching for optimal parameter values is extremely expensive computationally. Instead, we grossly approximate the optimal values by deterministically setting parameters based on the Viterbi parse of the training data parsed so far. We rely on the post-pass, described later, to refine parameter values.

Referring to the rules in Table 1, the parameter ϵ is set to an arbitrary small constant. The values of the parameters $p(A)$ are set to the (smoothed) frequency of the $X \rightarrow A$ reduction in the Viterbi parse of the data seen so far. The remaining symbols are set to expand uniformly among their possible expansions.

3.3 Constraining Moves

Consider the move of creating a rule of the form $A \rightarrow BC$. This corresponds to k^3 different specific rules that might be created, where k is the current number of symbols in the grammar. As it is too computationally expensive to consider each of these rules at every point in the search, we use heuristics to constrain which moves are appraised.

For the left-hand side of a rule, we always create a new symbol. This heuristic selects the optimal choice the vast majority of the time; however, under this constraint the moves described earlier in this section cannot yield arbitrary context-free languages. To partially address this, we add the move

Move 3: Create a rule of the form $A \rightarrow AB|B$

With this iteration move, we can construct grammars that generate arbitrary regular languages. As yet, we have not implemented moves that enable the construction of arbitrary context-free grammars; this belongs to future work.

To constrain the symbols we consider on the right-hand side of a new rule, we use what we call *triggers*.[3] A *trigger* is a phenomenon in the Viterbi parse of a sentence that is indicative that a particular move might lead to a better grammar. For example, in Figure 1 the fact that the symbols A_{talks} and A_{slowly} occur adjacently is indicative that it could be profitable to create a rule $B \rightarrow A_{talks}A_{slowly}$. We have developed a set of triggers for each move in our move set, and only consider a specific move if it is triggered in the sentence currently being parsed in the incremental processing.

3.4 Post-Pass

A conspicuous shortcoming in our search framework is that the grammars in our search space are fairly unexpressive. Firstly, recall that our grammars model a sentence as a sequence of independently generated symbols; however, in language there is a large dependence between adjacent constituents. Furthermore, the only free parameters in our search are the parameters $p(A)$; all other symbols (except S) are fixed to expand uniformly. These choices were necessary to make the search tractable.

To address this issue, we use an Inside-Outside algorithm post-pass. Our methodology is derived from that described by Lari and Young (1990). We create n new nonterminal symbols $\{X_1, \ldots, X_n\}$, and create all rules of the form:

$$
\begin{array}{lllll}
X_i & \rightarrow & X_j & X_k & i,j,k \in \{1,\ldots,n\} \\
X_i & \rightarrow & A & & i \in \{1,\ldots,n\}, \\
 & & & & A \in N_{old} - \{S, X\}
\end{array}
$$

N_{old} denotes the set of nonterminal symbols acquired in the initial grammar induction phase, and X_1 is taken to be the new sentential symbol. These new rules replace the first three rules listed in Table 1. The parameters of these rules are initialized randomly. Using this grammar as the starting point, we run the Inside-Outside algorithm on the training data until convergence.

In other words, instead of using the naive $S \rightarrow SX|X$ rule to attach symbols together in parsing data, we now use the X_i rules and depend on the Inside-Outside algorithm to train these randomly initialized rules intelligently. This post-pass allows us to express dependencies between adjacent symbols. In addition, it allows us to train parameters that were fixed during the initial grammar induction phase.

4 Previous Work

As mentioned, this work employs the Bayesian grammar induction framework described by Solomonoff (1960; 1964). However, Solomonoff does not specify a concrete search algorithm and only makes suggestions as to its nature.

[3] This is not to be confused with the use of the term *triggers* in dynamic language modeling.

Similar research includes work by Cook et al. (1976) and Stolcke and Omohundro (1994). This work also employs a heuristic search within a Bayesian framework. However, a different prior probability on grammars is used, and the algorithms are only efficient enough to be applied to small data sets.

The grammar induction algorithms most successful in language modeling include the Inside-Outside algorithm (Lari and Young, 1990; Lari and Young, 1991; Pereira and Schabes, 1992), a special case of the Expectation-Maximization algorithm (Dempster et al., 1977), and work by McCandless and Glass (1993). In the latter work, McCandless uses a heuristic search procedure similar to ours, but a very different search criteria. To our knowledge, neither algorithm has surpassed the performance of n-gram models in a language modeling task of substantial scale.

5 Results

To evaluate our algorithm, we compare the performance of our algorithm to that of n-gram models and the Inside-Outside algorithm.

For n-gram models, we tried $n = 1, \ldots, 10$ for each domain. For smoothing a particular n-gram model, we took a linear combination of all lower order n-gram models. In particular, we follow standard practice (Jelinek and Mercer, 1980; Bahl et al., 1983; Brown et al., 1992) and take the smoothed i-gram probability to be a linear combination of the i-gram frequency in the training data and the smoothed $(i-1)$-gram probability, that is,

$$\begin{aligned} p(w_0 | W = w_{i-1} \cdots w_{-1}) = \\ \lambda_{i,c(W)} \frac{c(W w_0)}{c(W)} + \\ (1 - \lambda_{i,c(W)}) p(w_0 | w_{i-2} \cdots w_{-1}) \end{aligned}$$

where $c(W)$ denotes the count of the word sequence W in the training data. The smoothing parameters $\lambda_{i,c}$ are trained through the Forward-Backward algorithm (Baum and Eagon, 1967) on held-out data. Parameters $\lambda_{i,c}$ are tied together for similar c to prevent data sparsity.

For the Inside-Outside algorithm, we follow the methodology described by Lari and Young. For a given n, we create a probabilistic context-free grammar consisting of all Chomsky normal form rules over the n nonterminal symbols $\{X_1, \ldots X_n\}$ and the given terminal symbols, that is, all rules

$$\begin{array}{lllll} X_i & \rightarrow & X_j \quad X_k & & i, j, k \in \{1, \ldots, n\} \\ X_i & \rightarrow & a & & i \in \{1, \ldots, n\}, a \in T \end{array}$$

where T denotes the set of terminal symbols in the domain. All parameters are initialized randomly. From this starting point, the Inside-Outside algorithm is run until convergence.

For smoothing, we combine the expansion distribution of each symbol with a uniform distribution, that is, we take the smoothed parameter $p_s(A \rightarrow \alpha)$ to be

$$p_s(A \rightarrow \alpha) = (1 - \lambda) p_u(A \rightarrow \alpha) + \lambda \frac{1}{n^3 + n|T|}$$

where $p_u(A \rightarrow \alpha)$ denotes the unsmoothed parameter. The value $n^3 + n|T|$ is the number of different ways a symbol expands under the Lari and Young methodology. The parameter λ is trained through the Inside-Outside algorithm on held-out data. This smoothing is also performed on the Inside-Outside post-pass of our algorithm. For each domain, we tried $n = 3, \ldots, 10$.

Because of the computational demands of our algorithm, it is currently impractical to apply it to large vocabulary or large training set problems. However, we present the results of our algorithm in three medium-sized domains. In each case, we use 4500 sentences for training, with 500 of these sentences held out for smoothing. We test on 500 sentences, and measure performance by the entropy of the test data.

In the first two domains, we created the training and test data artificially so as to have an ideal grammar in hand to benchmark results. In particular, we used a probabilistic grammar to generate the data. In the first domain, we created this grammar by hand; the grammar was a small English-like probabilistic context-free grammar consisting of roughly 10 nonterminal symbols, 20 terminal symbols, and 30 rules. In the second domain, we derived the grammar from manually parsed text. From a million words of parsed Wall Street Journal data from the Penn treebank, we extracted the 20 most frequently occurring symbols, and the 10 most frequently occurring rules expanding each of these symbols. For each symbol that occurs on the right-hand side of a rule but which was not one of the most frequent 20 symbols, we create a rule that expands that symbol to a unique terminal symbol. After removing unreachable rules, this yields a grammar of roughly 30 nonterminals, 120 terminals, and 160 rules. Parameters are set to reflect the frequency of the corresponding rule in the parsed corpus.

For the third domain, we took English text and reduced the size of the vocabulary by mapping each word to its part-of-speech tag. We used tagged Wall Street Journal text from the Penn treebank, which has a tag set size of about fifty.

In Tables 2–4, we summarize our results. The *ideal grammar* denotes the grammar used to generate the training and test data. For each algorithm, we list the best performance achieved over all n tried, and the *best n* column states which value realized this performance.

We achieve a moderate but significant improvement in performance over n-gram models and the

	best n	entropy (bits/word)	entr. relative to n-gram
ideal grammar		2.30	-6.5%
our algorithm	7	2.37	-3.7%
n-gram model	4	2.46	
Inside-Outside	9	2.60	$+5.7\%$

Table 2: English-like artificial grammar

	best n	entropy (bits/word)	entr. relative to n-gram
ideal grammar		4.13	-10.4%
our algorithm	9	4.44	-3.7%
n-gram model	4	4.61	
Inside-Outside	9	4.64	$+0.7\%$

Table 3: Wall Street Journal-like artificial grammar

Inside-Outside algorithm in the first two domains, while in the part-of-speech domain we are outperformed by n-gram models but we vastly outperform the Inside-Outside algorithm.

In Table 5, we display a sample of the number of parameters and execution time (on a Decstation 5000/33) associated with each algorithm. We choose n to yield approximately equivalent performance for each algorithm. The *first pass* row refers to the main grammar induction phase of our algorithm, and the *post-pass* row refers to the Inside-Outside post-pass.

Notice that our algorithm produces a significantly more compact model than the n-gram model, while running significantly faster than the Inside-Outside algorithm even though we use an Inside-Outside post-pass. Part of this discrepancy is due to the fact that we require a smaller number of new nonterminal symbols to achieve equivalent performance, but we have also found that our post-pass converges more quickly even given the same number of nonterminal symbols.

6 Discussion

Our algorithm consistently outperformed the Inside-Outside algorithm in these experiments. While we partially attribute this difference to using a Bayesian instead of maximum-likelihood objective function, we believe that part of this difference results from a more effective search strategy. In particular, though both algorithms employ a greedy hill-climbing strategy, our algorithm gains an advantage by being able to add new rules to the grammar.

In the Inside-Outside algorithm, the gradient descent search discovers the "nearest" local minimum in the search landscape to the initial grammar. If there are k rules in the grammar and thus k parameters, then the search takes place in a fixed k-dimensional space $\mathbb{R}^k$. In our algorithm, it is possible to expand the hypothesis grammar, thus increasing the dimensionality of the parameter space that is being searched. An apparent local minimum in the space $\mathbb{R}^k$ may no longer be a local minimum in the space $\mathbb{R}^{k+1}$; the extra dimension may provide a pathway for further improvement of the hypothesis grammar. Hence, our algorithm should be less prone to suboptimal local minima than the Inside-Outside algorithm.

Outperforming n-gram models in the first two domains demonstrates that our algorithm is able to take advantage of the grammatical structure present in data. However, the superiority of n-gram models in the part-of-speech domain indicates that to be competitive in modeling naturally-occurring data, it is necessary to model collocational information accurately. We need to modify our algorithm to more aggressively model n-gram information.

7 Conclusion

This research represents a step forward in the quest for developing grammar-based language models for natural language. We induce models that, while being substantially more compact, outperform n-gram language models in medium-sized domains. The algorithm runs essentially in time and space linear in the size of the training data, so larger domains are within our reach.

However, we feel the largest contribution of this work does not lie in the actual algorithm specified, but rather in its indication of the potential of the induction framework described by Solomonoff in 1964. We have implemented only a subset of the moves that we have developed, and inspection of our results gives reason to believe that these additional moves may significantly improve the performance of our algorithm.

Solomonoff's induction framework is not re-

	best n	entropy (bits/word)	entr. relative to n-gram
n-gram model	6	3.01	
our algorithm	7	3.15	+4.7%
Inside-Outside	7	3.93	+30.6%

Table 4: English sentence part-of-speech sequences

WSJ artif.	n	entropy (bits/word)	no. params	time (sec)
n-gram	3	4.61	15000	50
IO	9	4.64	2000	30000
first pass			800	1000
post-pass	5	4.60	4000	5000

Table 5: Parameters and Training Time

stricted to probabilistic context-free grammars. After completing the implementation of our move set, we plan to explore the modeling of context-sensitive phenomena. This work demonstrates that Solomonoff's elegant framework deserves much further consideration.

Acknowledgements

We are indebted to Stuart Shieber for his suggestions and guidance, as well as his invaluable comments on earlier drafts of this paper. This material is based on work supported by the National Science Foundation under Grant Number IRI-9350192 to Stuart M. Shieber.

References

D. Angluin and C.H. Smith. 1983. Inductive inference: theory and methods. *ACM Computing Surveys*, 15:237–269.

L.R. Bahl, J.K. Baker, P.S. Cohen, F. Jelinek, B.L. Lewis, and R.L. Mercer. 1978. Recognition of a continuously read natural corpus. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 422–424, Tulsa, Oklahoma, April.

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190, March.

J.K. Baker. 1975. The DRAGON system – an overview. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23:24–29, February.

J.K. Baker. 1979. Trainable grammars for speech recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*, pages 547–550, Boston, MA, June.

L.E. Baum and J.A. Eagon. 1967. An inequality with application to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematicians Society*, 73:360–363.

Peter F. Brown, Vincent J. DellaPietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands: North-Holland, May.

M.D. Kernighan, K.W. Church, and W.A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of the Thirteenth International Conference on Computational Linguistics*, pages 205–210.

K. Lari and S.J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

K. Lari and S.J. Young. 1991. Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 5:237–257.

Ming Li and Paul Vitányi. 1993. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag.

Michael K. McCandless and James R. Glass. 1993. Empirical acquisition of word and phrase classes in the ATIS domain. In *Third European Conference on Speech Communication and Technology*, Berlin, Germany, September.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracket corpora. In *Proceedings of the 30th Annual Meeting of the ACL*, pages 128–135, Newark, Delaware.

P. Resnik. 1992. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the 14th International Conference on Computational Linguistics.*

J. Rissanen. 1978. Modeling by the shortest data description. *Automatica*, 14:465–471.

Y. Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the 14th International Conference on Computational Linguistics.*

C.E. Shannon. 1951. Prediction and entropy of printed English. *Bell Systems Technical Journal*, 30:50–64, January.

R.J. Solomonoff. 1960. A preliminary report on a general theory of inductive inference. Technical Report ZTB-138, Zator Company, Cambridge, MA, November.

R.J. Solomonoff. 1964. A formal theory of inductive inference. *Information and Control*, 7:1–22, 224–254, March, June.

Rohini Srihari and Charlotte Baltus. 1992. Combining statistical and syntactic methods in recognizing handwritten sentences. In *AAAI Symposium: Probabilistic Approaches to Natural Language*, pages 121–127.