Natural Language Engineering **4** (3): 191–209. Printed in the United Kingdom

© 1998 Cambridge University Press

A fast method for statistical grammar induction

WIDE R. HOGENHOUT and YUJI MATSUMOTO

Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-01, Japan

(*Received 28 September 1998*)

Abstract

The statistical induction of stochastic context free grammars from bracketed corpora with the Inside Outside Algorithm is an appealing method for grammar learning, but the computational complexity of this algorithm has made it impossible to generate a large scale grammar. Researchers from natural language processing and speech recognition have suggested various methods to reduce the computational complexity and, at the same time, guide the learning algorithm towards a solution by, for example, placing constraints on the grammar. We suggest a method that strongly reduces that computational cost of the algorithm without placing constraints on the grammar. This method can in principle be combined with any of the constraints on grammars that have been suggested in earlier studies. We show that it is feasible to achieve results equivalent to earlier research, but with much lower computational effort. After creating a small grammar, the grammar is incrementally increased while rules that have become obsolete are removed at the same time. We explain the modifications to the algorithm, give results of experiments and compare these to results reported in other publications.

1 Introduction

The availability of large corpora creates the opportunity to model the structures humans recognize in sentences that appear in everyday natural language. One well known method for modelling such structures is the Inside Outside Algorithm, which was first described by Baker (1979).

The statistical induction of Stochastic Context Free Grammars (SCFG) has the attraction that it does not require any assumptions about the grammar that is being created, other than the basic assumption that SCFGs are a suitable model of the language. However, the main problem with this kind of modelling has been the computational complexity; the Inside Outside Algorithm requires $O(n^3|w|^3)$ of training time per sentence w for a grammar with n nonterminals, per iteration (Lari and Young 1990).

The algorithm has been used in two different applications. First, there is a line of research (Fujisaki, Jelinek, Cocke, Black and Nishino 1989; Black, Garside and Leech 1993; Hogenhout and Matsumoto 1996) that concentrates on the reestimation of hand written grammars. This has none or much less problems with time complexity since the structure of the grammar is already decided and the grammar usually

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

generates a limited number of parses for a given sentence. However, this does require the grammar to be developed by hand, a task that demands much expensive, highly skilled labour.

A second line of research concentrates on estimating a SCFG initially containing rules that may not be linguistically plausible, either from scratch (Lari and Young 1990; Lari and Young 1991; Pereira and Schabes 1992; Schabes, Roth and Osborne 1993), or by giving the grammar some initial structure without writing a complete grammar (Sharman, Jelinek and Mercer 1990; Briscoe and Waegner 1992).

In the experiments of Pereira and Schabes (1992) and Schabes *et al.* (1993), the algorithm was applied to sentences from a treebank. The brackets of the treebank were used to reduce the number of possible structures by disallowing any structure that crosses some treebank bracket. This greatly reduces training time, and Schabes *et al.* were able to parse short sentences of the *Wall Street Journal* Corpus with 90.2% accuracy using a 15-nonterminal grammar.

The extension to the algorithm and the experiments we present, aim at improving the efficiency of the induction of grammars from scratch. We were able to strongly reduce the computational cost of the training, and at the same time achieved equivalent results when parsing short sentences of the *Wall Street Journal* Corpus. We also achieved better results on longer sentences with a larger grammar.

We suggest a method in which the grammar is induced incrementally. We start with induction of a small grammar, and stepwise add nonterminals to this grammar until it reaches the desired size. In order to do this one nonterminal is removed from the smaller grammar and divided into two, so that a larger grammar is obtained. In this paper we describe the method in detail and report on the results obtained in experiments.

The remainder of this paper is organized as follows. Section 2 will give a review of SCFGs and the Inside Outside Algorithm. Section 3 discusses related literature. Section 4 discusses the method we propose. Section 5 describes experiments carried out with this method, and section 6 discusses the results. Section 7 contains a conclusion.

2 Review of SCFGs and the Inside Outside Algorithm

In this section we give a short review of SCFGs and the Inside Outside Algorithm. A context free grammar is a four-tuple $G = (V, T, S, R)$ where V is a set of nonterminal symbols, T is a set of terminal symbols, S is the start symbol and R is a set of productions. In this article the nonterminals are usually indicated as $X_1, X_2, ...$, but it is also required that $S \in V$. The terminal symbols are a fixed set $\{w^1, ..., w^{|T|}\}$, but in this article we usually consider a string $w_1 ... w_n$ that can be produced with the grammar, where every symbol w_i $(1 \leq i \leq n)$ in the string is an element of T. The productions are of the form $X \rightarrow \gamma$ where X is a nonterminal $(X \in V)$ and γ is a string of terminal and nonterminal symbols $(\gamma \in (V \cup T)^*)$.

A SCFG is a Context Free Grammar together with a set of probabilities associated with the productions of the grammar, written as $P(X \rightarrow \gamma)$. It is required that

$$\forall X \rightarrow \gamma \in R : 0 < P(X \rightarrow \gamma)$$

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

and

$$\forall X \in V : \sum_{\gamma : X \to \gamma \in R} P(X \to \gamma) = 1$$

to obtain a well defined probabilistic model.

In grammar induction the grammar is often restricted to a Chomsky Normal Form (CNF) grammar. This means that rules in R must be of the form $X_i \to X_j X_k$ or $X_i \to w^j$ ($X_i, X_j, X_k \in V$, $w^j \in T$). As there can be no more than two symbols on the right-hand side, the number of rules for a CNF grammar are finite. The number of rules of a complete CNF grammar (i.e. containing all admissible rules) is $|V|^3 + |V| \times |T|$.

A string $w_1 ... w_n$ can be derived from a context free grammar by starting with a string consisting of only the start symbol S and rewriting the nonterminal symbols in the string until the string only contains terminal symbols. A derivation has the form

$$S = \gamma_0 \overset{r1}{\Longrightarrow} \gamma_1 \overset{r2}{\Longrightarrow} \gamma_2 \ldots \overset{rm}{\Longrightarrow} \gamma_n = w_1 ... w_n,$$

where every r_l ($1 \leq l \leq m$) is a rule ($r_l \in R$), and the double arrow signifies that the rule r_l is applied to rewrite a particular nonterminal in string γ_{l-1}. If this is always the leftmost nonterminal, it is called a *leftmost derivation*.

The probability of a derivation t is defined for a SCFG as the product of the probabilities of the rules involved in each step,

$$P(t) = \prod_{i=1}^{m} P(r_i),$$

where r_i is the rule applied in step i of t.

The probability of a string $w_1 ... w_n$ is the sum of the probabilities of all leftmost derivations of the string. If we let $t : S \overset{t}{\Longrightarrow} w_1 ... w_n$ denote a leftmost derivation of $w_1 ... w_n$, then

$$P(w_1 ... w_n) = \sum_{t : S \overset{t}{\Longrightarrow} w1 ... wn} P(t),$$

and it can be shown that the sum of the probabilities of all leftmost derivations is equal to 1.

2.1 Inside Outside Algorithm

This section discusses the Inside Outside Algorithm for SCFGs in CNF. A more extensive discussion can be found in Lari and Young (1990).

Imagine that a sample set of strings $s_1 ... s_n$ and a SCFG in CNF with initial parameters (rule probabilities) that can produce those strings, are given. The Inside Outside Algorithm is a method to find parameters for the SCFG that maximize the probability that the SCFG would produce this set of strings. Being a special case of the Estimation Maximization (EM) Algorithm (Dempster, Laird and Rubin 1977), it must be supplied with a set of initial parameters and it improves these until a local maximum is reached.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

Given a string $w_1...w_n$, we consider the probability that some nonterminal X_p was rewritten into a substring $w_i...w_j$ $(1 \leq i \leq j \leq n)$. The *Inside Probability* of nonterminal X_p is defined as

$$I(X_p, i, j) = P(w_i...w_j | X_p \text{ covers } i...j),$$

which represents the probability of X_p being rewritten into the substring $w_i...w_j$, given that X_p is at the top of this subtree. It follows that $I(X_p, i, i) = P(X_p \rightarrow w_i)$. Based on the assumption that the probabilities of productions are independent it can be shown that

$$\begin{aligned} I(X_p, i, j) &= \sum_{q,r,k} P(X_p \rightarrow X_q X_r) \\ &\qquad P(w_i...w_k | X_q \text{ covers } i...k) \\ &\qquad P(w_{k+1}...w_j | X_r \text{ covers } k+1...j) \\ &= \sum_{q,r,k} P(X_p \rightarrow X_q X_r) I(X_q, i, k) I(X_r, k+1, j), \end{aligned}$$

where q and r are to be chosen such that $X_q \in V$ and $X_r \in V$, and k such that $i \leq k < j$. This recursive definition allows calculation from the bottom to the top of the tree. Since the string must be covered by the start symbol, it also follows that

$$I(S, 1, n) = P(w_1...w_n | S \text{ covers } 1...n) = P(w_1...w_n).$$

The *Outside Probability* is in a sense the opposite, as it signifies the probability of a nonterminal covering part of the string, together with the rest of the string. The Outside Probability of nonterminal X_p covering a phrase $w_i...w_j$ is defined as

$$O(X_p, i, j) = P(w_1...w_{i-1}, X_p \text{ covers } i...j, w_{j+1}...w_n)$$

and it follows that $O(S, 1, n) = 1$. The equation for the outside probability is more complex since, in a CNF grammar, a nonterminal can occur as the right child or as the left child of its parent. Once again using the assumption that probabilities of productions are independent it can be shown that

$$\begin{aligned} O(X_p, i, j) &= \sum_{q,r,k} O(X_q, k, j) P(X_q \rightarrow X_r X_p) I(X_r, k, i-1) + \\ &\qquad \sum_{q,r \neq p,l} O(X_q, i, l) P(X_q \rightarrow X_p X_r) I(X_r, j+1, l), \end{aligned}$$

where q and r are again chosen such that $X_q \in V$ and $X_r \in V$, and k such that $1 \leq k < i$, and l such that $j < l \leq n$. This is once again a recursive definition, but this time the calculation is carried out from the top to the bottom.

To measure the probability given to the corpus, the *cross entropy* is often used rather than calculating the probability directly, see Cover and Thomas (1991) for a general explanation about this measure. The cross entropy estimate of a SCFG G for a corpus C is given by

$$\hat{H}(C, G) = -\frac{\sum_{s \in C} log P(s)}{\sum_{s \in C} |s|},$$

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

where $P(s)$ is the probability that G gives to string s.

Based on these definitions, it can be argued that the estimated number of times a rule is used in a set of samples, according to the probability distributions of the grammar, is equal to

$$C(X_p \to X_q X_r) = \sum_{s=w1\ldots wn} \frac{1}{P(w_1 \ldots w_n)} \left(\sum_{1 \le i,j,k \le n} O(p,i,k) P(X_p \to X_q X_r) I(q,i,j) I(r,j+1,k) \right),$$

and, for every $w \in T$, when it is equal to a word w_i in the sample,

$$C(X_p \to w) = \sum_{s=w1\ldots wn} \frac{1}{P(w_1 \ldots w_n)} \left(\sum_{i=1}^{n} O(p,i,i) P(X_p \to w_i, w_i = w) \right).$$

The algorithm has the following structure:

1. Generate an initial grammar consisting of a complete SCFG grammar in CNF, or possibly a subset of this, given a fixed number of nonterminals (n) and a fixed number of terminal symbols (m):

$$X_i \to X_j X_k \qquad (1 \le i,j,k \le n)$$

$$X_i \to w_j \qquad (1 \le i \le n; 1 \le j \le m),$$

   where $w_j \in T$ and $X_i, X_j, X_k \in V$.
2. Give the grammar rules an initial distribution. This can be a random starting point, if no initial estimate is available.
3. Parse the sample strings and collect the count (C) values for every rule in the grammar.
4. Calculate the cross entropy estimate. If the difference with the previous iteration is lower than a preset threshold, stop.
5. Reestimate the rule probability for every rule $X \to \gamma \in R$ using the equation

$$\hat{P}(X \to \gamma) = \frac{C(X \to \gamma)}{\sum_{\beta : X \to \beta \in R} C(X \to \beta)}.$$

6. replace $P(X \to \gamma)$ with $\hat{P}(X \to \gamma)$ and go back to step 3.

After training it is possible to discard those rules that obtained a very low probability. The terminal symbols can represent lexical words, but when the vocabulary is large it is often restricted to symbols that represent parts-of-speech.

Lari and Young (1990) have pointed out that the algorithm is naturally suitable for parallel processing. The sample strings can be separated into a number of subsets and the counts obtained from these parts can be calculated independently and combined when ready. This does not reduce the amount of computation, but it does speed up the process.

3 Inducing statistical grammars

In this section we will discuss some of the existing research that relates to statistical grammar induction. In general, two modifications to the scheme have been applied

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

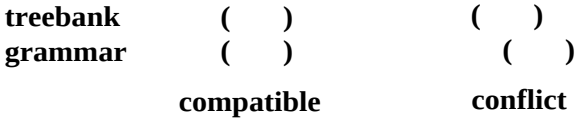


Fig. 1. Compatible and conflicting phrases.

to reduce the computational cost of the algorithm. First, it has been suggested to improve the algorithm by only applying those productions that do not conflict with interpretations given by humans. In this case, the algorithm does not only receive samples from the source, but also (partial) information about their derivations (section 3.1). Secondly, attempts to induce a better grammar were made by placing constraints on form of the rules allowed in the grammar (section 3.2) or gradually increasing the number of rules in the grammar (section 3.3). We also discuss a method that aimed at finding a good initial estimate, and a method for combining SCFGs (section 3.4).

3.1 Induction from bracketed corpora

The experiments reported in Pereira and Scabes (1992) showed that applying the Inside Outside Algorithm to a (partially) bracketed corpus has two advantages. First, it converges more quickly and strongly reduces the computational cost of the method. Secondly, it models the hierarchical structures of language, as indicated in the corpus, better. As was argued by Charniak and Carroll (1992b), in many cases the algorithm is likely to converge to a local maximum that does not reflect linguistical intuition, if such information is not provided.

The algorithm they use is characterized by the modified equations

$$I(X_p, i, j) = \bar{c}(i, j) \sum_{q,r,k} P(X_p \rightarrow X_q X_r) I(X_q, i, k) I(X_r, k, j)$$

and

$$O(X_p, i, j) = \bar{c}(i, j) \left(\sum_{q,r,k} O(X_q, r, k) P(X_q \rightarrow X_r X_p) I(X_r, k, i-1) + \sum_{q,r,k} O(X_q, p, r) P(X_q \rightarrow X_p X_r) I(X_r, j+1, k) \right),$$

which in both cases have been extended with the auxiliary function $\bar{c}(i, j)$. This function is 0 if the phrase $w_i ... w_j$ partially overlaps a treebank bracket and partially falls outside that bracket, and 1 otherwise. Figure 1 shows the way a conflict is defined, which is related to the crossing parentheses as given by Black *et al.* (1991). Pereira and Schabes argue that the recall component from the figure of merit given by Black *et al.* is not needed since the inference procedure is restricted to CNF grammars, and can therefore not avoid difficult decisions by leaving out brackets. In these experiments the terminal symbols of the grammar were part-of-speech symbols.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

Pereira and Schabes report on experiments showing that this enables the induction of a grammar that can accurately parse the ATIS corpus, but the linguistical simplicity of this corpus gave reason to doubt.

Schabes *et al.* (1993) report on a follow-up experiment with the same algorithm, now using the linguistically more complex *Wall Street Journal* Corpus. This proved that more complex structures can be learned in the same way. Sentences of up to 15 words were parsed with 90.2% bracketing accuracy using a 15-nonterminal grammar. Various sizes were tried for the training set, but this did not significantly affect the bracketing accuracy.

3.2 Enforcing structure in the grammar

Placing constraints on the form of the rules admitted in the grammar serves two goals. First, by reducing the number of rules involved in training the computational cost of grammar induction can be reduced. Secondly, if the constraints are correct, they will lead the algorithm in the right direction since undesirable rules are simply not present in the learning phase.

One approach is described by Sharman *et al.* (1990), although they do not train the grammar but instead use direct estimates from a treebank. They describe an experiment in which the grammar was in ID/LP format (Immediate Dominance and Linear Precedence), and received initial probabilities from the counts in a (fully bracketed) treebank. In this way, they restricted the form of the grammar, and at the same time separated the dominance probabilities from the precedence probabilities. The resulting grammar, containing of 100 terminal symbols and 16 nonterminal symbols, was tested on 42 sentences, but the small test set and the difference in test methods makes it impossible to compare the results to our approach.

An experiment with the Inside Outside Algorithm with restrictions on the grammar is described by Briscoe and Waegner (1992). The restrictions they place are similar to those in $\bar{X}$ theory. nonterminals that reflect major categories (such as N and V) have a bar level (zero, one or two), and are specified for noun and verb (e.g. a noun is classified +noun and -verb, an adverb is classified +noun and +verb). Each rule must be consistent with some constraints in order to be permitted as a part of the grammar. Most importantly, rules must be headed in the way specified by $\bar{X}$ theory. Briscoe and Waegner also give higher probabilities to what they call explicit rules to give the grammar more structure. These explicit rules form a hand-written grammar. This grammar is complemented by admissible, automatically generated rules called implicit rules that initially have a lower probability. They report that on a test set, 39 errors were made on a total of 189 rule applications (79% correct).

3.3 Incremental learning

The term incremental learning is used in the machine learning literature (Winston 1975; Reinke and Michalkski 1988; Gross 1988), and in cognitive science (Elman 1991). A related concept is batch learning (Klasner and Simon 1995). In the literature on machine learning the term is used in the sense that the number of samples

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

available to the machine learning algorithm gradually increases. The algorithm should therefore gradually improve performance by learning from new samples. Batch learning refers to learning in sets of data rather than one by one.

Carroll and Charniak (1992a; 1992b) describe a method for learning dependency grammars, using the concept of incremental learning. A dependency grammar can be considered as a special case of a CFG, and in that respect this method can be considered a combination of placing constraints on the rules in the grammar and incremental learning. In the study of Carroll and Charniak, the grammar is used to parse a corpus, and each time a sentence is encountered that can not be parsed with the grammar, all possible rules that are applicable to that sentence are added to the grammar. After each addition the grammar is retrained. Also, when the probability of a sentence becomes very low, the sentence is considered negative training material, and some of the relating rules are removed from the grammar. They report that in this way simple grammars could be induced. They did, however, not report on the accuracy of the analyses that a Viterbi-style parser would choose when it relies on these grammars.

We use the term *incremental induction* to refer to an increase in nonterminals (in other words, a change to the object that is being learned), and not in training data. Our approach should therefore be considered one of batch learning, and not incremental learning in the sense of increasing sample size.

Brown, Della Pietra, deSouza, Lai and Mercer (1992) describe a method that succeeds in extracting classes that reflect syntactically or semantically based groupings. This can be thought to be an incremental categorization method, but one which is based on co-occurrences and applied to bigrams, rather than the context free languages we are considering.

3.4 Other extensions

Lari and Young (1990) describe an experiment where the Inside Outside Algorithm is preceded with a pre-training algorithm. By first training a Hidden Markov Model (HMM) and converting the result to a statistical grammar, they managed to reduce the number of cycles needed for the Inside Outside Algorithm. In this way they reduced the computational cost significantly. A difference with our research is that the initial grammar they obtain is a regular language, since it was obtained with an HMM model, whereas our algorithm takes a smaller context free grammar rather than a regular grammar as starting point. As a future experiment, it may be interesting to experimentally compare their and our method of speeding up grammar learning.

Kita (1996) reports on a *mixture probabilistic CFG* approach. He shows how ideas from cluster-based language modelling can be applied to statistical grammar learning. By separating the corpus into clusters that represent different types of language (topics or domains), he shows how a topic-dependent language model can be created.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

Suppose the training corpus is divided into clusters $T_1 \ldots T_N$, and let $P_i(s)$ denote the probability of sentence s using the probabilities obtained by training the grammar with cluster T_i. The probability of sentence s is calculated by

$$P(S) = \sum_{i=1}^{N} P_i(S) P(s \text{ arises from cluster } T_i),$$

where, in the study by Kita, the probability of a sentence arising from a cluster is simply estimated on the basis of the size of the cluster. He shows that this significantly reduces the cross entropy of a test set. Note that this study was carried out using spoken language modelling, whereas our experiments are directed towards written language.

4 Incremental induction

The goal of this research is finding a way to induce grammars at less cost in terms of computational effort, but equivalent to those obtained by the original Inside Outside Algorithm. This does not only mean faster, a goal that can be attained by parallel processing. We seek for a modification of the algorithm that allows induction with less computational effort. As this makes it possible to use more nonterminals, we also intend to test whether the grammar can profit from this to parse longer sentences.

Our experiments are close to those performed by Schabes *et al.* (1993). We used the same number of nonterminals (15), the same corpus (*Wall Street Journal* Corpus) and about the same amount of training data, but we created the grammar in much less time than the regular algorithm requires.

Instead of starting with a grammar consisting of all possible rules for a given number of nonterminals, we start with a small number of nonterminals and gradually increase this to the desired number. At the same time we remove rules that have received a low probability so that the grammar grows much slower in terms of the number of rules.

This incremental induction scheme is based on the intuition that a small grammar can form the basis of a somewhat larger grammar. Suppose that a grammar with n nonterminals has been trained for a certain number of iterations. It is possible to take one nonterminal from this grammar, removing all rules that contain the nonterminal, and replace it with two new nonterminals. In a sense we split up a nonterminal into two, thus creating a grammar with $n + 1$ nonterminals.

More precisely, we take the following steps:

1. Select a nonterminal X_q that is being used in the grammar.
2. Remove all rules of the form $X_q \rightarrow X_j X_k$, $X_q \rightarrow t$, $X_j \rightarrow X_q X_k$, $X_k \rightarrow X_j X_q$, for every $X_j, X_k \in V$ and every $t \in T$.
3. Create all rules possible with the two nonterminals X_q and $X_{q'}$. In other words, all rules of the form $X_i \rightarrow X_j X_k$, where either X_i, X_j or X_k is X_q or $X_{q'}$. This includes multiple occurrences, so for example $X_q \rightarrow X_q X_q$ and $X_q \rightarrow X_{q'} X_q$ are also created.
4. Also, create the rules $X_q \rightarrow t$ and $X_{q'} \rightarrow t$ for every $t \in T$.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

5. Give these new rules randomized probabilities, so that their total probability mass equals that of nonterminal X_q before it was removed.

If no rules are deleted, this can be used to create a grammar that is completely equal in size to the grammar that would be induced directly from $n+1$ nonterminals. But we are interested in the possibility of removing those rules that received a very low probability, thereby maintaining a small grammar while the number of nonterminals increases.

The idea behind this process is that when the number of nonterminals is low (as it unvariably is) one nonterminal will take on various roles (represent various grammatical entities) and this will have a negative effect on the grammar. Separating one nonterminal into two and randomizing the related probabilities, i.e. destroying this part of the grammar, allows the algorithm to separate these roles while the rest of the grammar does not undergo major changes.

In theory it is also possible to separate more than one nonterminal at once, but this would strongly increase the computational cost and may well hamper convergence.

Selection One question we left open is the selection of a nonterminal: one needs some criterion to decide what nonterminal should undergo the split operation. In the experiments we are reporting on, we chose the nonterminal with the highest count as it is given by the Inside Outside Algorithm, in other words we used the nonterminal with the highest frequency in the training corpus.

The motivation is that this nonterminal has the most data available for training. Choosing another nonterminal may result in nonterminals that have too little training data to give meaningful estimates to their rules. However, there are other clues for choosing a nonterminal, and we almost never know if this is the best choice.

What we do know is that in our experiments this always was the best choice for the first split operation. We have tried several alternatives, such as taking the nonterminal that has the smallest Kullback–Leibler distance to the uniform distribution or the nonterminal that has the lowest count, but all of these these were less satisfying than the criterion mentioned above. Nevertheless, our criterion for selecting a nonterminal may not be optimal, and future work may identify situations in which other criteria render better results.

Amount of Rules If no rules would be removed before separating a nonterminal (step 2), the size of the grammar would simply be n^3. Our approach aims at reducing this number. To keep things simple, we allow kn^2 rules where k is what we call the *tolerance factor*. By setting k to some value we can decide how many rules the grammar will contain.

The procedure is as follows:

1. Train the grammar for a certain number of iterations.
2. Discard the rules of the form $X_i \rightarrow X_j X_k$ with the lowest probabilities until kn^2 such rules remain.
3. Separate one nonterminal, and repeat the process.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

The number of iterations is chosen to be such that the grammar somewhat stabilizes, but low enough to avoid unnecessary cycles. This corresponds to a higher threshold than is used for termination.

We have experimented with various values of k. Most of these experiments showed that the value of k does not have much influence on the results. When n is between 7 and 9, there is some negative effect if k is smaller than 4, so it was set at 4. When n became higher than 9, we discovered no significant decrease in performance when k was as low as 1. Only $k < 1$ gave an inferior performance. The size of the current experiments was not sufficient to give further results about the critical value of k, or about the possibility to restrict the grammar to kn rules.

Convergence Clearly, the algorithm is guaranteed to converge between separations, since the regular Inside Outside Algorithm is applied, but a separation is not guaranteed to improve the cross entropy estimate of the grammar in the long run.

This problem can in theory be overcome (to some extent) by generating for the two new nonterminals only the rules that existed for the selected nonterminal in the previous grammar. By also giving these the same distribution and only adding a small amount of noise (to make sure they start diverging) it is possible to make sure the new grammar has a cross entropy that is very close to the previous grammar.

Whereas this approach is theoretically more satisfying, we obtained disappointing results in practice (see also the section on experimental results). For this reason we continued with the mentioned approach, that holds the risk of degrading cross entropy.

5 The Experiments

This section first describes a small pilot experiment that serves to demonstrate and test the working of the separation process. Next, we continue with a larger experiment, using up to 15 nonterminals. Finally, we give results for a grammar with 18 nonterminals trained on sentences of up to 20 words.

In all experiments we use rules of Chomsky Normal Form. The terminal symbols of the grammar corresponded to the part-of-speech symbols in the corpus. The initial distribution was the uniform distribution, with a small amount of random noise to make sure that the nonterminals start diverging. [1] We use both the *cross entropy* and the *bracketing accuracy* of the grammar for evaluation purposes. Cross entropy shows to what extent the grammar has stabilized, but it does not always give a good indication of how well it models the corpus. For this reason, we also use a Viterbi parser that selects the most likely parse for test sentences and calculates the bracketing accuracy of the grammar, i.e. the proportion of phrases in those parses that are compatible with the treebank bracketings in the test set.

[1] To our knowledge, there is no reason to assume that this distribution gives a better result than other initial distributions.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

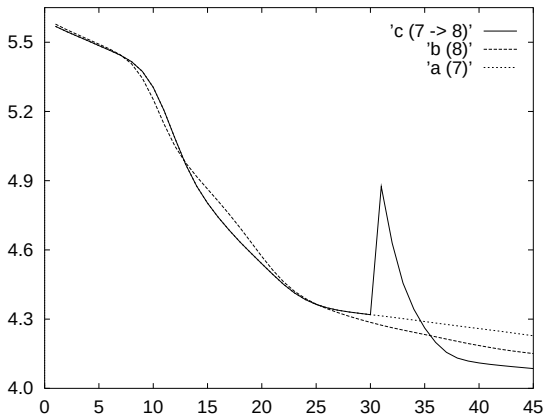


Fig. 2. Entropy values for pilot experiment.

5.1 Pilot experiment

We first conducted a pilot experiment to answer the following question: Can the entropy of the grammar be kept at the same level when a nonterminal is separated in two, can it improve, or will it deteriorate? This experiment is only meant to study the viability of the approach.

We compared three grammars:

(a) a grammar with seven nonterminals (regular procedure, no separations);
(b) a grammar with eight nonterminals (regular procedure, no separations);
(c) a grammar with seven nonterminals (same initial distribution as (a)) where one nonterminal was separated, turning it into an 8 nonterminal grammar.

Figure 2 shows the results in cross entropy of the three grammars. Grammar (b) has more nonterminals than (a), so it takes more time to train (this is not visible in the figure) and achieves a lower cross entropy. During the first 30 iterations grammars (a) and (c) are equal, then (c) suddenly increases enormously at the point where a nonterminal is separated.

This is caused by the loss of information about the distribution of one nonterminal. Since the two new nonterminals receive random distributions the cross entropy of the grammar suddenly deteriorates. However, this soon improves and the final entropy is actually lower than grammar (b), which had eight nonterminals from the beginning.

This experiment has been repeated many times, and always gives the same pattern for one separation. It shows that separating nonterminals can be a good idea, but it does not show that the entropy will be lower than a grammar that was trained with more nonterminals from the start after a number of separations. In our experience, although it improves at most separations, it occasionally remains about equal.

5.2 Experiment with 15 nonterminals

We proceeded to perform an experiment with the *Wall Street Journal* treebank. The value for m (number of parts of speech) was 31. This is lower than the 47 in the

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

experiment reported on by Schabes *et al* (1993) as some similar low frequency parts of speech were merged. The reason for merging some parts of speech is that they occur so infrequently that relating rules can never be estimated reliably. Some of these did not even occur in the training data, making it impossible to estimate related rules. As they also occurred very infrequently in the test data, the effect on the final result can be considered small. The training set consisted of 1000 sentences of up to 15 words and tests were performed on 100 sentences that were not used in training.

The experiment started with a 7-nonterminal grammar. This was trained for 30 iterations. After this the nonterminal with the highest count was separated into two and the resulting grammar was trained for 15 iterations. This was repeated until the grammar had 15 nonterminals, each time training the grammar for 15 iterations after separating a nonterminal into two.

At first, after training the grammar for 30 or 15 iterations and before separating a nonterminal, the number of rules of the form $X_i \rightarrow X_j X_k$ was reduced to $4n^2$. (When the number of rules is not reduced there would be n^3 such rules.) When the number of nonterminals became 10, the number of rules was further reduced to only n^2 (before each separation). Since we do one more deletion at the end, the final grammar had n^2 rules of the form $X_i \rightarrow X_j X_k$. Table 1 shows the steps that are taken in order to induce a 15-nonterminal grammar.

The rules that were removed, were simply those with the smallest probability (irrespective of the nonterminals they contained). Theoretically, this could result in nonterminals becoming obsolete when their rules are removed, but we never encountered this problem. It appears that, when the grammar is retrained after a separation, the part that was not randomized does not undergo major changes during the subsequent training.

The final 15-nonterminal grammar we induced thus had 225 rules of the type $X_i \rightarrow X_j X_k$. We also limited the number of rules of the type $X_i \rightarrow t_j$, so that it remained at 7 for each part of speech. The final number of rules was therefore $225 + 7 * 31 = 442$.

5.3 Experiment with 18 nonterminals

The last experiment is very similar to the second one, but this time we trained with sentences of up to 20 words and expanded the grammar to 18 nonterminals. The procedure is almost identical to that used for the 15-nonterminal grammar. Besides using longer sentences and continuing to 18 nonterminals, we let $k = 4$ for the grammar when it had 7–9 nonterminals, $k = 3$ when it had 10–14 nonterminals, and $k = 1$ when it had 15 or more nonterminals.

6 Results

The result in entropy on test data and training data for the second experiment, trained and tested on sentences of up to 15 words, and using up to 15 nonterminals, is indicated in figure 3. The peaks in this figure represent the points where some nonterminal is separated into two. This includes randomizing the probabilities of

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

Table 1. *Steps of induction process*

1	Generate complete grammar with 7 nonterminals
2	Train 7-nonterminal grammar for 30 cycles
3	Remove rules of type $N_1 \rightarrow N_2N_3$ with lowest probability until 4×7^2 such rules remain
4	Remove the rule of type $N \rightarrow w$ with the lowest probability for every terminal symbol w.
5	Select nonterminal with highest count and carry out split
6	Train 8-nonterminal grammar for 15 cycles
7	Remove rules of type $N_1 \rightarrow N_2N_3$ with lowest probability until 4×8^2 such rules remain
8	Remove the rule of type $N \rightarrow w$ with the lowest probability for every terminal symbol w.
9	Select nonterminal with highest count and carry out split
10	Train 9-nonterminal grammar for 15 cycles
11	Remove rules of type $N_1 \rightarrow N_2N_3$ with lowest probability until 4×9^2 such rules remain
12	Remove the rule of type $N \rightarrow w$ with the lowest probability for every terminal symbol w.
13	Select nonterminal with highest count and carry out split
14	Train 10-nonterminal grammar for 15 cycles
15	Remove rules of type $N_1 \rightarrow N_2N_3$ with lowest probability until 10^2 such rules remain
16	Remove the rule of type $N \rightarrow w$ with the lowest probability for every terminal symbol w.
17	Select nonterminal with highest count and carry out split
18	Train 11-nonterminal grammar for 15 cycles
...	...
31	Remove rules of type $N_1 \rightarrow N_2N_3$ with lowest probability until 14^2 such rules remain
32	Remove the rule of type $N \rightarrow w$ with the lowest probability for every terminal symbol w.
33	Select nonterminal with highest count and carry out split
34	Train 15-nonterminal grammar for 15 cycles
35	Remove rules of type $N_1 \rightarrow N_2N_3$ with lowest probability until 15^2 such rules remain
36	Remove the rule of type $N \rightarrow w$ with the lowest probability for every terminal symbol w.

the rules involved, and therefore leads to an upward jump in entropy. But after this severe loss the entropy values become lower than they would have become with less nonterminals almost every time. For example, the 7-nonterminal grammar comes to an entropy of about 4.3, even after running many more iterations than are indicated here, whereas the final 15-nonterminal grammar came to about 3.6. This shows that our method gradually improves the grammar, even though a large part of the grammar is discarded before each separation.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

Table 2. *Comparison of bracketing accuracies*

Length	0–10	0–15	10–19	20–30
Induced grammar (15) (%)	92.0	91.7	83.8	72.0
Induced grammar (18) (%)	94.1	91.5	86.9	81.8
Schabes *et al.* (%)	94.4	90.2	82.5	71.5
Right linear (%)	76	70	63	50
Treebank grammar (%)	46	31	25	

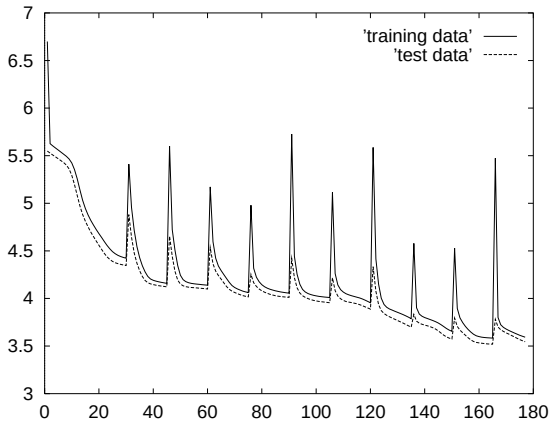


Fig. 3. Plot of entropy values for training data and test data (y-axis) against number of iterations (x-axis). Maximum of 15 words and 15 nonterminals.

Figure 5 gives the same data for the third experiment, trained and tested on sentences of up to 20 words, and using up to 18 nonterminals. The shape is quite similar, although it has a higher number of peaks as more nonterminals-splits were carried out. It is remarkable that the final entropy, about 3.5, is even slightly less than those of the smaller grammar while the training and test sentences are longer.

Table 2 compares the results with those of the grammar induced in the experiments reported by Schabes *et al.* (1993), with a grammar that produces a right-branching

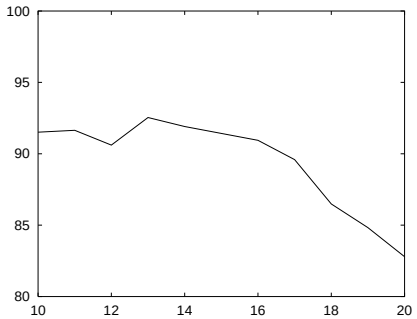


Fig. 4. Bracket accuracy (y-axis) against maximum sentence length (x-axis) for 15 nonterminals.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

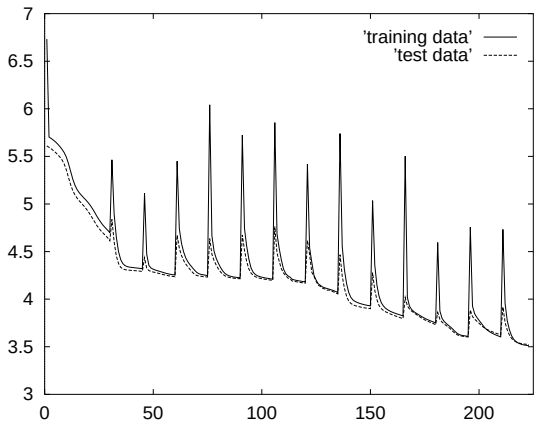


Fig. 5. Plot of entropy values for training data and test data (y-axis) against number of iterations (x-axis). Maximum of 20 words and 18 nonterminals.

structure for everything except the final punctuation, and with a grammar directly abstracted from the corpus. (These figures were taken from Schabes *et al.* (1993).) This shows that the 15-nonterminal grammar we induced scores equivalent to the grammar induced in Schabes *et al.* (1993). The 18-nonterminal grammar does not give further improvement on the shorter sentences, but gives a substantial improvement to sentences of 10–19 words, and especially to those with 20–30 words. This shows that the 18-nonterminal grammar has learned substantially more about the structure of longer sentences.

The alternative approach that was mentioned briefly in section 4, which only duplicates existing rules at a separation and only introduces a slight amount of noise, did not obtain an accuracy of more than 82% on sentences of up to 15 words.

Figure 4 shows the accuracies of the 15-nonterminal grammar on the test set, differentiated by sentence length. Please note that this is cumulative; for example 20 on the x-axis means 'shorter than 20 words'. Figure 6 shows the same accuracies for the 18-nonterminal grammar. This again shows that by increasing the number of nonterminals, the size of the sentence can be increased strongly while maintaining the same accuracy.

The number of iterations is much larger than other experiments reported in the literature, but the actual training time is much shorter. This was expected since the the grammar grows at a speed in the order of $O(n^2)$ rather than $O(n^3)$. In the implementation we used, and on the computer we used for run the experiment, the procedure we have described was about 20 times faster than the regular procedure for a 15-nonterminal grammar, even though the regular procedure only requires about 60 cycles. This shows that by using the shape of a small grammar, it is feasible to construct a larger grammar much faster, without loss of quality. The number of cycles becomes higher, but the computational effort is much lower.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

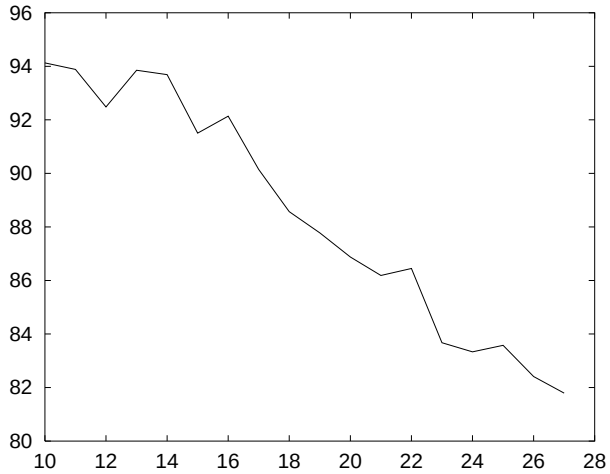


Fig. 6. Bracket accuracy (y-axis) against maximum sentence length (x-axis) for 18 nonterminals.

7 Conclusion

We have presented a new method for statistical induction of SCFGs. The discussion of earlier modifications of the method used for induction shows that other attempts to reduce the computational cost coincide with placing constraints on the grammar, occasionally combined with an incremental approach. Some of these methods also aim at obtaining a grammar in a particular form, such as a grammar with $\bar{\text{X}}$ constraints or a dependency grammar.

Our method aims purely at reducing computational cost. This is realized with an incremental strategy that places no constraints on the grammar. Experiments show that it is feasible to obtain a grammar of equivalent quality in much less time when using this strategy. The size of the grammar is allowed to grow with speed $O(n^2)$ instead of $O(n^3)$, and this directly translates to less computational effort in parsing sentences and a smaller number of parses.

The essential part of the technique is using information from a smaller grammar to reduce the amount of training that must be performed on a larger grammar. For this goal we destroy part of the smaller grammar, but leave the major part in tact, to retain some of the obtained elements and at the same time allow the larger grammar to be adjusted.

The experiments show that, with necessary adjustments, running the Inside Outside Algorithm is feasible with less computational effort than before, and this makes it possible to obtain larger and better grammars. In principle, this method can be combined with any of the constraints on the grammar rules that were suggested in earlier studies, such as $\bar{\text{X}}$-style grammars or dependency grammars. We expect that this result will also apply to language modelling for speech recognition systems.

If future research were to result in a grammar where the number of nonterminals becomes closer to the number of nonterminals used in grammars written by human experts, it may be possible to make a comparison between the concepts used by induced and written grammars.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

Acknowledgements

We would like to express our appreciation to the reviewers. This article has benefited greatly from their valuable comments and criticisms.

References

Baker, J. K. (1979) Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, D. H. Klatt and J. J. Wolf (eds.), pp. 547–550.

Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B. and Strzalkowski, T. (1991) A procedure for quantitatively comparing the syntactic coverage of English grammars. *Proceedings of the Workshop on Speech and Natural Language, Defense Advanced Research Projects Agency, U.S. Govt*., pp. 306–311.

Black, E., Garside, R. and Leech, G. (1993) *Statistically-Driven Computer Grammars of English: The IBM/Lancaster Approach*. Rodopi.

Briscoe, T. and Waegner, N. (1992) Robust stochastic parsing using the inside-outside algorithm. *Workshop Notes, Statistically-Based NLP Techniques*, AAAI, pp. 33–41.

Brown, P. F., Della Pietra, S. A., deSouza, P. V., Lai, J. C. and Mercer, R. L. (1992) Class-based n-gram models of natural language. *Computational Linguistics*, **18**(4): 467–479.

Carroll, G. and Charniak, E. (1992*a*) Learning probabilistic dependency grammars from labelled text. *Working Notes, Fall Symposium Series*, AAAI, pp. 25–32.

Carroll, G. and Charniak, E. (1992*b*) Two experiments on learning probabilistic dependency grammars from corpora. *Workshop Notes, Statistically-Based Natural Language Programming Techniques*, AAAI, pp. 1–13.

Cover, T. M. and Thomas, J. A. (1991) *Elements of Information Theory*. Wiley.

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. B*, **39**: 1–38.

Elman, J. L. (1991) Incremental learning, or the importance of starting small. *Proceedings 13th Annual Conference of the Cognitive Science Society*, pp. 443–448.

Fujisaki, T., Jelinek, F., Cocke, J., Black, E. and Nishino, T. (1989) A probabilistic method for sentence disambiguation. *Proceedings of the 1st International Workshop on Parsing Technologies*, pp. 105–114.

Gross, P. K. (1988) Incremental multiple concept learning using experiments. *Proceedings 5th International Conference on Machine Learning*.

Hogenhout, W. R. and Matsumoto, Y. (1996) Training stochastical grammars on semantical categories. In *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, E. Riloff, S. Wermter and G. Scheler (eds.), pp. 160–172. Springer-Verlag.

Kita, K. (1996) Mixture probabilistic context-free grammar. *J. Natural Language Processing*, **3**(4): 103–113.

Klasner, N. and Simon, H. U. (1995) From noise-free to noise-tolerant and from on-line to batch learning. *Proceedings 8th Annual Conference on Computational Learning Theory*, pp. 250–257.

Lari, K. and Young, S. J. (1990) The estimation of stochastic context-free grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, **4**: 35–56.

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

Pereira, F. and Schabes, Y. (1992) Inside Outside reestimation from partially bracketed corpora. *Proceedings 30th Annual Meeting of the Association for Computational Linguistics*, pp. 128–135.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press

Reinke, R. E. and Michalkski, R. S. (1988) Incremental learning of decision rules: A method and experimental results. In *Machine Intelligence*, J. E. Hayes, D. Michie and J. Richards (eds.). Oxford University Press.

Schabes, Y., Roth, M. and Osborne, R. (1993) Parsing the Wall Street Journal with the Inside-Outside algorithm. *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 341–347.

Sharman, R., Jelinek, F. and Mercer, R. (1990) Generating a grammar for statistical training. *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 267–274.

Winston, P. H. (1975) Learning structural descriptions from examples. In *The Psychology of Computer Vision*, P. H. Winston (ed.). McGraw Hill.

https://doi.org/10.1017/S1351324998001983 Published online by Cambridge University Press