Probabilistic Context-Free Grammar Induction Based on Structural Zeros

Mehryar Mohri
Courant Institute of Mathematical Sciences
and Google Research
251 Mercer Street
New York, NY 10012
mohri@cs.nyu.edu

Brian Roark
Center for Spoken Language Understanding
OGI at Oregon Health & Science University
20000 NW Walker Road
Beaverton, Oregon 97006
roark@cslu.ogi.edu

Abstract

We present a method for induction of concise and accurate probabilistic context-free grammars for efficient use in early stages of a multi-stage parsing technique. The method is based on the use of statistical tests to determine if a non-terminal combination is unobserved due to sparse data or hard syntactic constraints. Experimental results show that, using this method, high accuracies can be achieved with a non-terminal set that is orders of magnitude smaller than in typically induced probabilistic context-free grammars, leading to substantial speed-ups in parsing. The approach is further used in combination with an existing reranker to provide competitive WSJ parsing results.

1 Introduction

There is a very severe speed vs. accuracy tradeoff in stochastic context-free parsing, which can be explained by the grammar factor in the running-time complexity of standard parsing algorithms such as the CYK algorithm (Kasami, 1965; Younger, 1967). That algorithm has complexity $O(n^3|P|)$, where n is the length in words of the sentence parsed, and $|P|$ is the number of grammar productions. Grammar non-terminals can be split to encode richer dependencies in a stochastic model and improve parsing accuracy. For example, the parent of the left-hand side (LHS) can be annotated onto the label of the LHS category (Johnson, 1998), hence differentiating, for instance, between expansions of a VP with parent S and parent VP. Such annotations, however, tend to substantially increase the number of grammar productions as well as the ambiguity of the grammar, thereby significantly slowing down the parsing algorithm. In the case of bilexical grammars, where categories in binary grammars are annotated with their lexical heads, the grammar factor contributes an additional $O(n^2|V_D|^3)$ complexity, leading to an overall $O(n^5|V_D|^3)$ parsing complexity, where $|V_D|$ is the number of delexicalized non-terminals (Eisner, 1997). Even with special modifications to the basic CYK algorithm, such as those presented by Eisner and Satta (1999), improvements to the stochastic model are obtained at the expense of efficiency.

In addition to the significant cost in efficiency, increasing the non-terminal set impacts parameter estimation for the stochastic model. With more productions, much fewer observations per production are available and one is left with the hope that a subsequent smoothing technique can effectively deal with this problem, regardless of the number of non-terminals created. Klein and Manning (2003b) showed that, by making certain linguistically-motivated node label annotations, but avoiding certain other kinds of state splits (mainly lexical annotations) models of relatively high accuracy can be built without resorting to smoothing. The resulting grammars were small enough to allow for exhaustive CYK parsing; even so, parsing speed was significantly impacted by the state splits: the test-set parsing time reported was about 3s for average length sentences, with a memory usage of 1GB.

This paper presents an automatic method for deciding which state to split in order to create concise and accurate unsmoothed probabilistic context-free grammars (PCFGs) for *efficient* use in early stages of a multi-stage parsing technique. The method is based on the use of statistical tests to determine if a non-terminal combination is unobserved due to the limited size of the sample (*sampling zero*) or because it is grammatically impossible (*structural zero*). This helps introduce a relatively small number of new non-terminals with little additional parsing

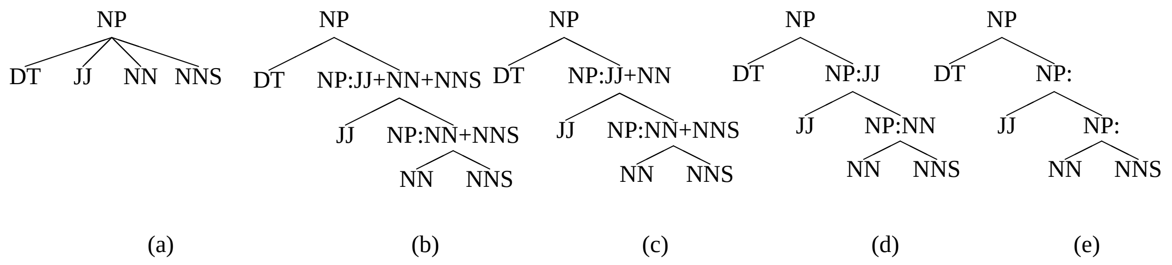

Figure 1: Five representations of an n-ary production, $n = 4$. (a) Original production (b) Right-factored production (c) Right-factored Markov order-2 (d) Right-factored Markov order-1 (e) Right-factored Markov order-0

overhead. Experimental results show that, using this method, high accuracies can be achieved with orders of magnitude fewer non-terminals than in typically induced PCFGs, leading to substantial speed-ups in parsing. The approach can further be used in combination with an existing reranker to provide competitive WSJ parsing results.

The remainder of the paper is structured as follows. Section 2 gives a brief description of PCFG induction from treebanks, including non-terminal label-splitting, factorization, and relative frequency estimation. Section 3 discusses the statistical criteria that we explored to determine structural zeros and thus select non-terminals for the factored PCFG. Finally, Section 4 reports the results of parsing experiments using our exhaustive k-best CYK parser with the concise PCFGs induced from the Penn WSJ treebank (Marcus et al., 1993).

2 Grammar induction

A context-free grammar $G = (V, T, S^\dagger, P)$, or CFG in short, consists of a set of non-terminal symbols V, a set of terminal symbols T, a start symbol $S^\dagger \in V$, and a set of production P of the form: $A \rightarrow \alpha$, where $A \in V$ and $\alpha \in (V \cup T)^*$. A PCFG is a CFG with a probability assigned to each production. Thus, the probabilities of the productions expanding a given non-terminal sum to one.

2.1 Smoothing and factorization

PCFGs induced from the Penn Treebank have many productions with long sequences of non-terminals on the RHS. Probability estimates of the RHS given the LHS are often smoothed by making a Markov assumption regarding the conditional independence of a category on those more than k categories away (Collins, 1997; Charniak, 2000):

$$\begin{aligned}\mathrm{P}(X \rightarrow Y_1...Y_n) &= \mathrm{P}(Y_1|X)\prod_{i=2}^{n}\mathrm{P}(Y_i|X, Y_1\cdots Y_{i-1})\\ &\approx \mathrm{P}(Y_1|X)\prod_{i=2}^{n}\mathrm{P}(Y_i|X, Y_{i-k}\cdots Y_{i-1}).\end{aligned}$$

Making such a Markov assumption is closely related to grammar transformations required for certain efficient parsing algorithms. For example, the CYK parsing algorithm takes as input a Chomsky Normal Form PCFG, i.e., a grammar where all productions are of the form $X \rightarrow Y Z$ or $X \rightarrow a$, where X, Y, and Z are non-terminals and a a terminal symbol.[1]. Binarized PCFGs are induced from a treebank whose trees have been factored so that n-ary productions with $n > 2$ become sequences of $n-1$ binary productions. Full right-factorization involves concatenating the final $n-1$ categories from the RHS of an n-ary production to form a new composite non-terminal. For example, the original production NP → DT JJ NN NNS shown in Figure 1(a) is factored into three binary rules, as shown in Figure 1(b). Note that a PCFG induced from such right-factored trees is weakly equivalent to a PCFG induced from the original treebank, i.e., it describes the same language.

From such a factorization, one can make a Markov assumption for estimating the production probabilities by simply recording only the labels of the first k children dominated by the composite factored label. Figure 1 (c), (d), and (e) show right-factored trees of Markov orders 2, 1 and 0 respectively.[2] In addition to being used for smoothing

[1]Our implementation of the CYK algorithm has been extended to allow for unary productions with non-terminals on the RHS in the PCFG.

[2]Note that these factorizations do not provide exactly the stated Markov order for all dependencies in the productions, because we are restricting factorization to only produce binary productions. For example, in Figure 1(e), the probability of the

PCFG	Time (s)	Words/s	$\|V\|$	$\|P\|$	LR	LP	F
Right-factored	4848	6.7	10105	23220	69.2	73.8	71.5
Right-factored, Markov order-2	1302	24.9	2492	11659	68.8	73.8	71.3
Right-factored, Markov order-1	445	72.7	564	6354	68.0	73.0	70.5
Right-factored, Markov order-0	206	157.1	99	3803	61.2	65.5	63.3
Parent-annotated, Right-factored, Markov order-2	7510	4.3	5876	22444	76.2	78.3	77.2

Table 1: Baseline results of exhaustive CYK parsing using different probabilistic context-free grammars. Grammars are trained from sections 2-21 of the Penn WSJ Treebank and tested on all sentences of section 24 (no length limit), given weighted k-best POS-tagger output. The second and third columns report the total parsing time in seconds and the number of words parsed per second. The number of non-terminals, $|V|$, is indicated in the next column. The last three columns show the labeled recall (LR), labeled precision (LP), and F-measure (F).

as mentioned above, these factorizations reduce the size of the non-terminal set, which in turn improves CYK efficiency. The efficiency benefit of making a Markov assumption in factorization can be substantial, given the reduction of both non-terminals and productions, which improves the grammar constant. With standard right-factorization, as in Figure 1(b), the non-terminal set for the PCFG induced from sections 2-21 of the Penn WSJ Treebank grows from its original size of 72 to 10105, with 23220 productions. With a Markov factorization of orders 2, 1 and 0 we get non-terminal sets of size 2492, 564, and 99, and rule production sets of 11659, 6354, and 3803, respectively.

These reductions in the size of the non-terminal set from the original factored grammar result in an order of magnitude reduction in complexity of the CYK algorithm. One common strategy in statistical parsing is what can be termed an approximate coarse-to-fine approach: a simple PCFG is used to prune the search space to which richer and more complex models are applied subsequently (Charniak, 2000; Charniak and Johnson, 2005). Producing a "coarse" chart as efficiently as possible is thus crucial (Charniak et al., 1998; Blaheta and Charniak, 1999), making these factorizations particularly useful.

2.2 CYK parser and baselines

To illustrate the importance of this reduction in non-terminals for efficient parsing, we will present baseline parsing results for a development set. For these baseline trials, we trained a PCFG on sections 2-21 of the Penn WSJ Treebank (40k sentences, 936k words), and evaluated on section 24 (1346 sentences, 32k words). The parser takes as input the weighted k-best POS-tag sequences of a perceptron-trained tagger, using the tagger documented in Hollingshead et al. (2005). The number of tagger candidates k for all trials reported in this paper was $0.2n$, where n is the length of the string. From the weighted k-best list, we derive a conditional probability of each tag at position i by taking the sum of the exponential of the weights of all candidates with that tag at position i (softmax).

The parser is an exhaustive CYK parser that takes advantage of the fact that, with the grammar factorization method described, factored non-terminals can only occur as the second child of a binary production. Since the bulk of the non-terminals result from factorization, this greatly reduces the number of possible combinations given any two cells. When parsing with a parent-annotated grammar, we use a version of the parser that also takes advantage of the partitioning of the non-terminal set, i.e., the fact that any given non-terminal has already its parent indicated in its label, precluding combination with any non-terminal that does not have the same parent annotated.

Table 1 shows baseline results for standard right-factorization and factorization with Markov orders 0-2. Training consists of applying a particular grammar factorization to the treebank prior to inducing a PCFG using maximum likelihood (relative frequency) estimation. Testing consists of exhaustive CYK parsing of all sentences in the development set (no length limit) with the induced grammar, then detransforming the maximum likelihood parse back to the original format for evaluation against the reference parse. Evaluation includes the standard PARSEVAL measures labeled precision (LP) and labeled recall (LR), plus the harmonic mean (F-measure) of these two scores. We also present a result using parent annotation (Johnson, 1998) with a 2nd-order Markov assumption. Parent annotation occurs prior to treebank factorization. This condition is roughly equivalent to the $h = 1, v = 2$ in Klein and Manning

final NNS depends on the preceding NN, despite the Markov order-0 factorization. Because of our focus on efficient CYK, we accept these higher order dependencies rather than producing unary productions. Only n-ary rules n>2 are factored.

(2003b)[3].

From these results, we can see the large efficiency benefit of the Markov assumption, as the size of the non-terminal and production sets shrink. However, the efficiency gains come at a cost, with the Markov order-0 factored grammar resulting in a loss of a full 8 percentage points of F-measure accuracy. Parent annotation provides a significant accuracy improvement over the other baselines, but at a substantial efficiency cost.

Note that the efficiency impact is not a strict function of either the number of non-terminals or productions. Rather, it has to do with the number of competing non-terminals in cells of the chart. Some grammars may be very large, but less ambiguous in a way that reduces the number of cell entries, so that only a very small fraction of the productions need to be applied for any pair of cells. Parent annotation does just the opposite – it increases the number of cell entries for the same span, by creating entries for the same constituent with different parents. Some non-terminal annotations, e.g., splitting POS-tags by annotating their lexical items, result in a large grammar, but one where the number of productions that will apply for any pair of cells is greatly reduced.

Ideally, one would obtain the efficiency benefit of the small non-terminal set demonstrated with the Markov order-0 results, while encoding key grammatical constraints whose absence results in an accuracy loss. The method we present attempts to achieve this by using a statistical test to determine *structural zeros* and modifying the factorization to remove the probability mass assigned to them.

3 Detecting Structural Zeros

The main idea behind our method for detecting structural zeros is to search for events that are individually very frequent but that do not co-occur. For example, consider the Markov order-0 binary rule production in Figure 2. The production NP→NP NP: may be very frequent, as is the NP:→CC NN production, but they never co-occur together, because NP does not conjoin with NN in the Penn Treebank. If the counts of two such events a and b, e.g., NP→NP NP: and NP:→CC NN are very large, but the count of their co-occurrence

[3]Their Markov order-2 factorization does not follow the linear order of the children, but rather includes the head-child plus one other, whereas our factorization does not involve identification of the head child.

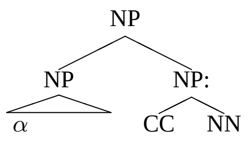

Figure 2: Markov order-0 local tree, with possible non-local ¡state-split information.

is zero, then the co-occurrence of a and b can be viewed as a candidate for the list of events that are structurally inadmissible. The probability mass for the co-occurrence of a and b can be removed by replacing the factored non-terminal NP: with NP:CC:NN whenever there is a CC and an NN combining to form a factored NP non-terminal.

The expansion of the factored non-terminals is not the only event that we might consider. For example, a frequent left-most child of the first child of the production, or a common left-corner POS or lexical item, might never occur with certain productions. For example, 'SBAR→IN S' and 'IN→of' are both common productions, but they never co-occur. We focus on left-most children and left-corners because of the factorization that we have selected, but the same idea could be applied to other possible state splits.

Different statistical criteria can be used to compare the counts of two events with that of their co-occurrence. This section examines several possible criteria that are presented, for ease of exposition, with general sequences of events. For our specific purpose, these sequences of events would be two rule productions.

3.1 Notation

This section describes several statistical criteria to determine if a sequence of two events should be viewed as a structural zero. These tests can be generalized to longer and more complex sequences, and to various types of events, e.g., word, word class, or rule production sequences.

Given a corpus $\mathcal{C}$, and a vocabulary Σ, we denote by c_a the number of occurrences of a in $\mathcal{C}$. Let n be the total number of observations in $\mathcal{C}$. We will denote by $\bar{a}$ the set $\{b \in \Sigma : b \neq a\}$. Hence $c_{\bar{a}} = n - c_a$. Let $\mathrm{P}(a) = \frac{c_a}{n}$, and for $b \in \Sigma$, let $\mathrm{P}(a|b) = \frac{c_{ab}}{c_b}$. Note that $c_{\bar{a}b} = c_b - c_{ab}$.

3.2 Mutual information

The mutual information between two random variables X and Y is defined as

$$I(X;Y) = \sum_{x,y} \mathrm{P}(x,y) \log \frac{\mathrm{P}(x,y)}{\mathrm{P}(x)\mathrm{P}(y)}. \quad (1)$$

For a particular event sequence of length two ab, this suggests the following statistic:

$$\begin{aligned} I(ab) &= \log \mathrm{P}(ab) - \log \mathrm{P}(a) - \log \mathrm{P}(b) \\ &= \log c_{ab} - \log c_a - \log c_b + \log n \end{aligned}$$

Unfortunately, for $c_{ab} = 0$, $I(ab)$ is not finite. If we assume, however, that all unobserved sequences are given some ϵ count, then when $c_{ab} = 0$,

$$I(ab) = K - \log c_a - \log c_b, \quad (2)$$

where K is a constant. Since we need these statistics only for ranking purposes, we can ignore the constant factor.

3.3 Log odds ratio

Another statistic that, like mutual information, is ill-defined with zeros, is the *log odds ratio*:

$$\log(\hat{\theta}) = \log c_{ab} + \log c_{\bar{a}\bar{b}} - \log c_{\bar{a}b} - \log c_{a\bar{b}}.$$

Here again, if $c_{ab} = 0$, $\log(\hat{\theta})$ is not finite. But, if we assign to all unobserved pairs a small count ϵ, when $c_{ab} = 0$, $c_{\bar{a}b} = c_b$, and the expression becomes

$$\log(\hat{\theta}) = K + \log c_{\bar{a}\bar{b}} - \log c_b - \log c_a. \quad (3)$$

3.4 Pearson chi-squared

For any $i, j \in \Sigma$, define $\hat{\mu}_{ij} = \frac{c_i c_j}{n}$. The Pearson chi-squared test of independence is then defined as follows:

$$\mathcal{X}^2 = \sum_{\substack{i \in \{a,\bar{a}\} \\ j \in \{b,\bar{b}\}}} \frac{(c_{ij} - \hat{\mu}_{ij})^2}{\hat{\mu}_{ij}} = \sum_{\substack{i \in \{a,\bar{a}\} \\ j \in \{b,\bar{b}\}}} \frac{(nc_{ij} - c_i c_j)^2}{nc_i c_j}.$$

In the case of interest for us, $c_{ab} = 0$ and the statistic simplifies to:

$$\mathcal{X}^2 = \frac{c_a c_b}{n} + \frac{c_a^2 c_b}{nc_{\bar{a}}} + \frac{c_a c_b^2}{nc_{\bar{b}}} + \frac{c_a^2 c_b^2}{nc_{\bar{a}} c_{\bar{b}}} = \frac{nc_a c_b}{c_{\bar{a}} c_{\bar{b}}}. \quad (4)$$

3.5 Log likelihood ratio

Pearson's chi-squared statistic assumes a normal or approximately normal distribution, but that assumption typically does not hold for the occurrences of rare events (Dunning, 1994). It is then preferable to use the likelihood ratio statistic which allows us to compare the null hypothesis, that $\mathrm{P}(b) = \mathrm{P}(b|a) = \mathrm{P}(b|\bar{a}) = \frac{c_b}{n}$, with the hypothesis that $\mathrm{P}(b|a) = \frac{c_{ab}}{c_a}$ and $\mathrm{P}(b|\bar{a}) = \frac{c_{\bar{a}b}}{c_{\bar{a}}}$. In words, the null hypothesis is that the context of event a does not change the probability of seeing b. These discrete conditional probabilities follow a binomial distribution, hence the likelihood ratio is

$$\lambda = \frac{B[\mathrm{P}(b), c_{ab}, c_a]\ B[\mathrm{P}(b), c_{\bar{a}b}, c_{\bar{a}}]}{B[\mathrm{P}(b|a), c_{ab}, c_a]\ B[\mathrm{P}(b|\bar{a}), c_{\bar{a}b}, c_{\bar{a}}]}, \quad (5)$$

where $B[p, x, y] = p^x (1-p)^{y-x} \binom{y}{x}$. In the special case where $c_{ab} = 0$, $\mathrm{P}(b|\bar{a}) = \mathrm{P}(b)$, and this expression can be simplified as follows:

$$\begin{aligned} \lambda &= \frac{(1-\mathrm{P}(b))^{c_a} \mathrm{P}(b)^{c_{\bar{a}b}} (1-\mathrm{P}(b))^{c_{\bar{a}} - c_{\bar{a}b}}}{\mathrm{P}(b|\bar{a})^{c_{\bar{a}b}} (1-\mathrm{P}(b|\bar{a}))^{c_{\bar{a}} - c_{\bar{a}b}}} \\ &= (1-\mathrm{P}(b))^{c_a}. \end{aligned} \quad (6)$$

The log-likelihood ratio, denoted by G^2, is known to be asymptotically $\mathcal{X}^2$-distributed. In this case,

$$G^2 = -2c_a \log(1 - \mathrm{P}(b)), \quad (7)$$

and with the binomial distribution, it has has one degree of freedom, thus the distribution will have asymptotically a mean of one and a standard deviation of $\sqrt{2}$.

We experimented with all of these statistics. While they measure different ratios, empirically they seem to produce very similar rankings. For the experiments reported in the next section, we used the log-likelihood ratio because this statistic is well-defined with zeros and is preferable to the Pearson chi-squared when dealing with rare events.

4 Experimental results

We used the log-likelihood ratio statistic G^2 to rank unobserved events ab, where $a \subset P$ and $b \in V$. Let V_o be the original, unfactored non-terminal set, and let $\alpha \in (V_o\ :)^*$ be a sequence of zero or more non-terminal/colon symbol pairs. Suppose we have a frequent factored non-terminal $X{:}\alpha B$ for $X, B \in V_o$. Then, if the set of productions $X \rightarrow Y X{:}\alpha A$ with

$A \in V_o$ is also frequent, but $X \rightarrow Y X{:}\alpha B$ is unobserved, this is a candidate structural zero. Similar splits can be considered with non-factored non-terminals.

There are two state split scenarios we consider in this paper. Scenario 1 is for factored non-terminals, which are always the second child of a binary production. For use in Equation 7,

$$\begin{aligned} c_a &= \sum_{A \in V_o} c(X \rightarrow Y X{:}\alpha A) \\ c_b &= c(X{:}\alpha B) \quad \text{for } B \in V_o \\ c_{ab} &= c(X \rightarrow Y X{:}\alpha B) \\ \mathrm{P}(b) &= \frac{c(X{:}\alpha B)}{\sum_{A \in V_o} c(X{:}\alpha A)}. \end{aligned}$$

Scenario 2 is for non-factored non-terminals, which we will split using the leftmost child, the left-corner POS-tag, and the left-corner lexical item, which are easily incorporated into our grammar factorization approach. In this scenario, the non-terminal to be split can be either the left or right child in the binary production. Here we show the counts for the left child case for use in Equation 7:

$$\begin{aligned} c_a &= \sum_{A} c(X \rightarrow Y[\alpha A] Z) \\ c_b &= c(\mathrm{Y}[\alpha B]) \\ c_{ab} &= c(X \rightarrow Y[\alpha B] Z) \\ \mathrm{P}(b) &= \frac{c(Y[\alpha B])}{\sum_{A} c(Y[\alpha A])} \end{aligned}$$

In this case, the possible splits are more complicated than just non-terminals as used in factoring. Here, the first possible split is the left child category, along with an indication of whether it is a unary production. One can further split by including the left-corner tag, and even further by including the left-corner word. For example, a unary S category might be split as follows: first to S[1:VP] if the single child of the S is a VP; next to S[1:VP:VBD] if the left-corner POS-tag is VBD; finally to S[1:VP:VBD:went] if the VBD verb was 'went'.

Note that, once non-terminals are split by annotating such information, the base non-terminals, e.g., S, implicitly encode contexts other than the ones that were split.

Table 2 shows the unobserved rules with the largest G^2 score, along with the ten non-terminals that these productions suggest for inclusion in our non-terminal set. The highest scoring unobserved production is PP → IN[that] NP. It receives such a high score because the base production (PP → IN NP) is very frequent, and so is 'IN→that', but they jointly never occur, since 'IN→that' is a complementizer. This split non-terminal also shows up in the second-highest ranked zero, an SBAR with 'that' complementizer and an S child that consists of a unary VP. The unary S→VP production is very common, but never with a 'that' complementizer in an SBAR.

Unobserved production (added NT(s) in bold)	G^2 score
PP → **IN[that]** NP	7153.1
SBAR → IN[that] **S[1:VP]**	5712.1
SBAR → **IN[of]** S	5270.5
SBAR → **WHNP[1:WDT] S[1:VP:TO]**	4299.9
VP → AUX **VP[MD]**	3972.1
SBAR → **IN[in]** S	3652.1
NP → NP **VP[VB]**	3236.2
NP → NN **NP:CC:NP**	2796.3
SBAR → WHNP **S[1:VP:VBG]**	2684.9

Table 2: Top ten non-terminals to add, and the unobserved productions leading to their addition to the non-terminal set.

Note that the fourth-ranked production uses two split non-terminals. The fifth ranked rule presumably does not add much information to aid parsing disambiguation, since the AUX MD tag sequence is unlikely[4]. The eighth ranked production is the first with a factored category, ruling out coordination between NN and NP.

Before presenting experimental results, we will mention some practical issues related to the approach described. First, we independently parameterized the number of factored categories to select and the number of non-factored categories to select. This was done to allow for finer control of the amount of splitting of non-terminals of each type. To choose 100 of each, every non-terminal was assigned the score of the highest scoring unobserved production within which it occurred. Then the 100 highest scoring non-terminals of each type were added to the base non-terminal list, which originally consisted of the atomic treebank non-terminals and Markov order-0 factored non-terminals.

Once the desired non-terminals are selected, the training corpus is factored, and non-terminals are split if they were among the selected set. Note, how-

[4]In fact, we do not consider splits when both siblings are POS-tags, because these are unlikely to carry any syntactic disambiguation.

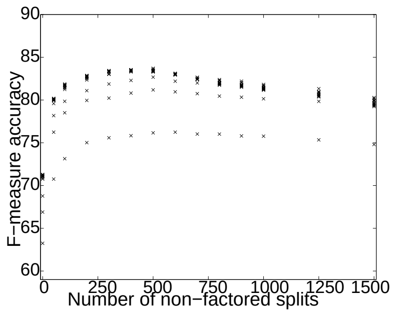

Figure 3: F-measure accuracy on development set versus the number of non-factored splits for the given run. Points represent different numbers of factored splits.

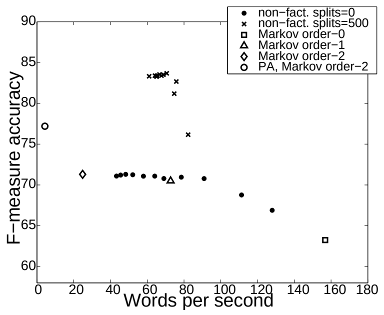

Figure 4: F-measure accuracy versus words-per-second for (1) no non-factored splits (i.e., only factored categories selected); (2) 500 non-factored splits, which was the best performing; and (3) four baseline results.

ever, that some of the information in a selected non-terminal may not be fully available, requiring some number of additional splits. Any non-terminal that is required by a selected non-terminal will be selected itself. For example, suppose that NP:CC:NP was chosen as a factored non-terminal. Then the second child of any local tree with that non-terminal on the LHS must either be an NP or a factored non-terminal with at least the first child identified as an NP, i.e., NP:NP. If that factored non-terminal was not selected to be in the set, it must be added. The same situation occurs with left-corner tags and words, which may be arbitrarily far below the category.

After factoring and selective splitting of non-terminals, the resulting treebank corpus is used to train a PCFG. Recall that we use the k-best output of a POS-tagger to parse. For each POS-tag and lexical item pair from the output of the tagger, we reduce the word to lower case and check to see if the combination is in the set of split POS-tags, in which case we split the tag, e.g., IN[that].

Figure 3 shows the F-measure accuracy for our trials on the development set versus the number of non-factored splits parameterized for the trial. From this plot, we can see that 500 non-factored splits provides the best F-measure accuracy on the dev set. Presumably, as more than 500 splits are made, sparse data becomes more problematic. Figure 4 shows the development set F-measure accuracy versus the number of words-per-second it takes to parse the development set, for non-factored splits of 0 and 500, at a range of factored split parameterizations. With 0 non-factored splits, efficiency is substantially impacted by increasing the factored splits, whereas it can be seen that with 500 non-factored splits, that impact is much less, so that the best performance is reached with both relatively few factored non-terminal splits, and a relatively small efficiency impact. The non-factored splits provide substantial accuracy improvements at relatively small efficiency cost.

Table 3 shows the 1-best and reranked 50-best results for the baseline Markov order-2 model, and the best-performing model using factored and non-factored non-terminal splits. We present the efficiency of the model in terms of words-per-second over the entire dev set, including the longer strings (maximum length 116 words)[5]. We used the k-best decoding algorithm of Huang and Chiang (2005) with our CYK parser, using on-demand k-best back-pointer calculation. We then trained a MaxEnt reranker on sections 2-21, using the approach outlined in Charniak and Johnson (2005), via the publicly available reranking code from that paper.[6] We used the default features that come with that package. The processing time in the table includes the time to parse and rerank. As can be seen from the trials, there is some overhead to these processes, but the time is still dominated by the base parsing.

We present the k-best results to demonstrate the benefits of using a better model, such as the one we have presented, for producing candidates for downstream processing. Even with severe pruning to only the top 50 candidate parses per string, which results in low oracle and reranked accuracy for the Markov order-2 model, the best-performing model based on structural zeros achieves a relatively high oracle accuracy, and reaches 88.0 and 87.5 percent F-measure accuracy on the dev (f24) and eval (f23) sets respectively. Note that the well-known Char-

[5]The parsing time with our model for average length sentences (23-25 words) is 0.16 seconds per sentence.

[6]`http://www.cog.brown.edu/˜mj/code`.

Technique	No. of Cands	Development (f24) Time(s)	Words/s	Oracle F	LR	LP	F	Eval (f23) LR	LP	F
Baseline, Markov order-2	1	1302	24.9	71.3	68.8	73.8	71.3	68.9	73.9	71.4
	50	1665	19.4	86.2	79.7	83.3	81.5	80.5	84.0	82.2
NT splits: factored=200	1	491	65.9	83.7	83.1	84.3	83.7	82.4	83.4	82.9
non-factored=500	50	628	51.5	93.8	87.4	88.7	88.0	87.1	88.0	87.5

Table 3: Parsing results on the development set (f24) and the evaluation set (f23) for the baseline Markov order-2 model and the best-performing structural zero model, with 200 factored and 500 non-factored non-terminal splits. 1-best results, plus reranking using a trained version of an existing reranker with 50 candidates.

niak parser (Charniak, 2000; Charniak and Johnson, 2005) uses a Markov order-3 baseline PCFG in the initial pass, with a best-first algorithm that is run past the first parse to populate the chart for use by the richer model. While we have demonstrated exhaustive parsing efficiency, our model could be used with any of the efficient search best-first approaches documented in the literature, from those used in the Charniak parser (Charniak et al., 1998; Blaheta and Charniak, 1999) to A* parsing (Klein and Manning, 2003a). By using a richer grammar of the sort we present, far fewer edges would be required in the chart to include sufficient quality candidates for the richer model, leading to further downstream savings of processing time.

5 Conclusion

We described a method for creating concise PCFGs by detecting structural zeros. The resulting unsmoothed PCFGs have far higher accuracy than simple induced PCFGs and yet are very efficient to use. While we focused on a small number of simple non-terminal splits that fit the factorization we had selected, the technique presented is applicable to a wider range of possible non-terminal annotations, including head or parent annotations. More generally, the ideas and method for determining structural zeros (vs. sampling zeros) can be used in other contexts for a variety of other learning tasks.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant IIS-0447214. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. The first author's work was partially funded by the New York State Office of Science Technology and Academic Research (NYSTAR).

References

D. Blaheta and E. Charniak. 1999. Automatic compensation for parser figure-of-merit flaws. In *Proceedings of ACL*, pages 513–518.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*, pages 173–188.

E. Charniak, S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the 6th Workshop on Very Large Corpora*, pages 127–133.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.

M.J. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*, pages 16–23.

T. Dunning. 1994. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1):61–74.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of ACL*, pages 457–464.

J. Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of the International Workshop on Parsing Technologies*, pages 54–65.

K. Hollingshead, S. Fisher, and B. Roark. 2005. Comparing and combining finite-state and context-free parsers. In *Proceedings of HLT-EMNLP*, pages 787–794.

L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pages 53–64.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):617–636.

T. Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report, AFCRL-65-758, Air Force Cambridge Research Lab., Bedford, MA.

D. Klein and C. Manning. 2003a. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of HLT-NAACL*.

D. Klein and C. Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of ACL*.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

D.H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.