Efficient Algorithms for Parsing the DOP Model *

Joshua Goodman
Harvard University
33 Oxford St.
Cambridge, MA 02138
goodman@das.harvard.edu

Abstract

Excellent results have been reported for Data-Oriented Parsing (DOP) of natural language texts (Bod, 1993c). Unfortunately, existing algorithms are both computationally intensive and difficult to implement. Previous algorithms are expensive due to two factors: the exponential number of rules that must be generated and the use of a Monte Carlo parsing algorithm. In this paper we solve the first problem by a novel reduction of the DOP model to a small, equivalent probabilistic context-free grammar. We solve the second problem by a novel deterministic parsing strategy that maximizes the expected number of correct constituents, rather than the probability of a correct parse tree. Using the optimizations, experiments yield a 97% crossing brackets rate and 88% zero crossing brackets rate. This differs significantly from the results reported by Bod, and is comparable to results from a duplication of Pereira and Schabes's (1992) experiment on the same data. We show that Bod's results are at least partially due to an extremely fortuitous choice of test data, and partially due to using cleaner data than other researchers.

Introduction

The Data-Oriented Parsing (DOP) model has a short, interesting, and controversial history. It was introduced by Remko Scha (1990), and was then studied by Rens Bod. Unfortunately, Bod (1993c, 1992) was not able to find an efficient exact algorithm for parsing using the model; however he did discover and implement Monte Carlo approximations. He tested these algorithms on a cleaned up version of the ATIS corpus, and achieved some very exciting results, reportedly getting 96% of his test set exactly correct, a huge improvement over previous results. For instance, Bod (1993b) compares these results to Schabes (1993), in which, for short sentences, 30% of the sentences have no crossing brackets (a much easier measure than exact match). Thus, Bod achieves an extraordinary 8-fold error rate reduction.

Not surprisingly, other researchers attempted to duplicate these results, but due to a lack of details of the parsing algorithm in his publications, these other researchers were not able to confirm the results (Magerman, Lafferty, personal communication). Even Bod's thesis (Bod, 1995a) does not contain enough information to replicate his results.

Parsing using the DOP model is especially difficult. The model can be summarized as a special kind of Stochastic Tree Substitution Grammar (STSG): given a bracketed, labelled training corpus, let *every* subtree of that corpus be an elementary tree, with a probability proportional to the number of occurrences of that subtree in the training corpus. Unfortunately, the number of trees is in general exponential in the size of the training corpus trees, producing an unwieldy grammar.

In this paper, we introduce a reduction of the DOP model to an exactly equivalent Probabilistic Context Free Grammar (PCFG) that is linear in the number of nodes in the training data. Next, we present an algorithm for parsing, which returns the parse that is expected to have the largest number of correct constituents. We use the reduction and algorithm to parse held out test data, comparing these results to a replication of Pereira and

* I would like to acknowledge support from National Science Foundation Grant IRI-9350192 and a National Science Foundation Graduate Student Fellowship. I would also like to thank Rens Bod, Stan Chen, Andrew Kehler, David Magerman, Wheeler Ruml, Stuart Shieber, and Khalil Sima'an for helpful discussions, and comments on earlier drafts, and the comments of the anonymous reviewers.

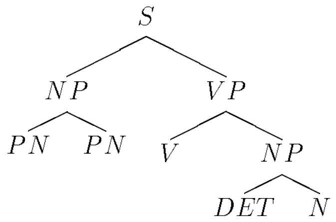


Figure 1: Training corpus tree for DOP example

Schabes (1992) on the same data. These results are disappointing: the PCFG implementation of the DOP model performs about the same as the Pereira and Schabes method. We present an analysis of the runtime of our algorithm and Bod's. Finally, we analyze Bod's data, showing that some of the difference between our performance and his is due to a fortuitous choice of test data.

This paper contains the first published replication of the full DOP model, i.e. using a parser which sums over derivations. It also contains algorithms implementing the model with significantly fewer resources than previously needed. Furthermore, for the first time, the DOP model is compared on the same data to a competing model.

Previous Research

The DOP model itself is extremely simple and can be described as follows: for every sentence in a parsed training corpus, extract every subtree. In general, the number of subtrees will be very large, typically exponential in sentence length. Now, use these trees to form a Stochastic Tree Substitution Grammar (STSG). There are two ways to define a STSG: either as a Stochastic Tree Adjoining Grammar (Schabes, 1992) restricted to substitution operations, or as an extended PCFG in which entire trees may occur on the right hand side, instead of just strings of terminals and nonterminals.

Given the tree of Figure 1, we can use the DOP model to convert it into the STSG of Figure 2. The numbers in parentheses represent the probabilities. These trees can be combined in various ways to parse sentences.

In theory, the DOP model has several advantages over other models. Unlike a PCFG, the use of trees allows capturing large contexts, making the model more sensitive. Since every subtree is included, even trivial ones corresponding to rules in a PCFG, novel sentences with unseen contexts

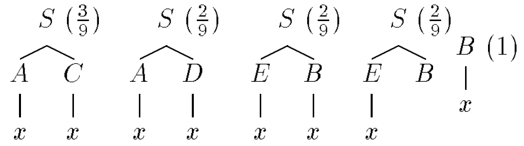
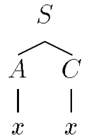


Figure 3: Simple Example STSG

can still be parsed.

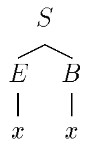
Unfortunately, the number of subtrees is huge; therefore Bod randomly samples 5% of the subtrees, throwing away the rest. This significantly speeds up parsing.

There are two existing ways to parse using the DOP model. First, one can find the most probable *derivation*. That is, there can be many ways a given sentence could be derived from the STSG. Using the most probable derivation criterion, one simply finds the most probable way that a sentence could be produced. Figure 3 shows a simple example STSG. For the string xx, what is the most probable derivation? The parse tree



has probability $\frac{3}{9}$ of being generated by the trivial derivation containing a single tree. This tree corresponds to the most probable derivation of xx.

One could try to find the most probable parse *tree*. For a given sentence and a given parse tree, there are many different derivations that could lead to that parse tree. The probability of the parse tree is the sum of the probabilities of the derivations. Given our example, there are two different ways to generate the parse tree



each with probability $\frac{2}{9}$, so that the parse tree has probability $\frac{4}{9}$. This parse tree is most probable.

Bod (1993c) shows how to approximate this most probable parse using a Monte Carlo algorithm. The algorithm randomly samples possible derivations, then finds the tree with the most sampled derivations. Bod shows that the most probable parse yields better performance than the most probable derivation on the exact match criterion.

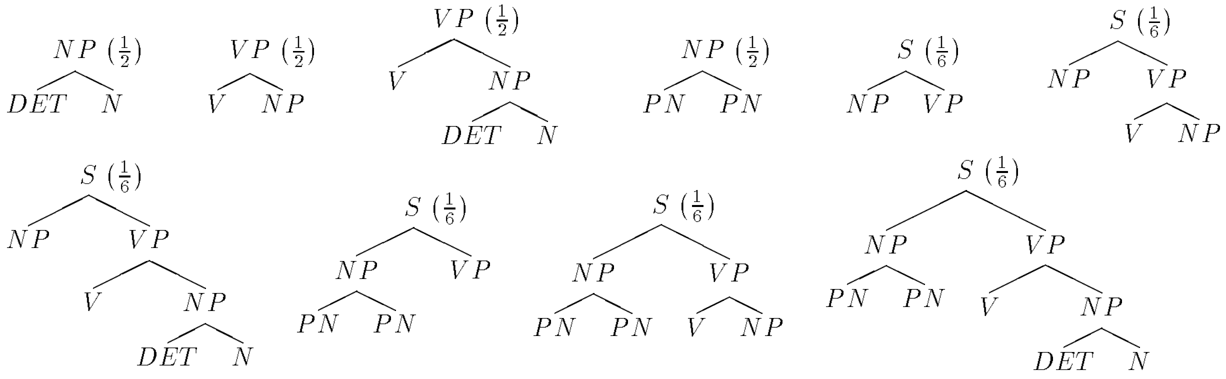


Figure 2: Sample STSG Produced from DOP Model

Khalil Sima'an (1996) implemented a version of the DOP model, which parses efficiently by limiting the number of trees used and by using an efficient most probable derivation model. His experiments differed from ours and Bod's in many ways, including his use of a different version of the ATIS corpus; the use of word strings, rather than part of speech strings; and the fact that he did not parse sentences containing unknown words, effectively throwing out the most difficult sentences. Furthermore, Sima'an limited the number of substitution sites for his trees, effectively using a subset of the DOP model.

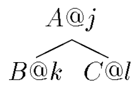
Reduction of DOP to PCFG

Unfortunately, Bod's reduction to a STSG is extremely expensive, even when throwing away 95% of the grammar. Fortunately, it is possible to find an equivalent PCFG that contains exactly eight PCFG rules for each node in the training data; thus it is $O(n)$. Because this reduction is so much smaller, we do not discard any of the grammar when using it. The PCFG is equivalent in two senses: first it generates the same strings with the same probabilities; second, using an isomorphism defined below, it generates the same trees with the same probabilities, although one must sum over several PCFG trees for each STSG tree.

To show this reduction and equivalence, we must first define some terminology. We assign every node in every tree a unique number, which we will call its address. Let $A@k$ denote the node at address k, where A is the non-terminal labeling that node. We will need to create one new non-terminal for each node in the training data. We will call this non-terminal A_k. We will call non-terminals of this form "interior" non-terminals, and the original non-terminals in the parse trees "exterior".

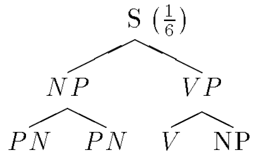
Let a_j represent the number of subtrees headed by the node $A@j$. Let a represent the number of subtrees headed by nodes with non-terminal A, that is $a = \sum_j a_j$.

Consider a node $A@j$ of the form:



How many subtrees does it have? Consider first the possibilities on the left branch. There are b_k non-trivial subtrees headed by $B@k$, and there is also the trivial case where the left node is simply B. Thus there are $b_k + 1$ different possibilities on the left branch. Similarly, for the right branch there are $c_l + 1$ possibilities. We can create a subtree by choosing any possible left subtree and any possible right subtree. Thus, there are $a_j = (b_k + 1)(c_l + 1)$ possible subtrees headed by $A@j$. In our example tree of Figure 1, both noun phrases have exactly one subtree: $np_4 = np_2 = 1$; the verb phrase has 2 subtrees: $vp_3 = 2$; and the sentence has 6: $s_1 = 6$. These numbers correspond to the number of subtrees in Figure 2.

We will call a PCFG subderivation isomorphic to a STSG tree if the subderivation begins with an external non-terminal, uses internal non-terminals for intermediate steps, and ends with external non-terminals. For instance, consider the tree



taken from Figure 2. The following PCFG subderivation is isomorphic: $S \Rightarrow \ NP@1 \ VP@2 \Rightarrow$

$PN \;\; PN \;\; VP@2 \Rightarrow PN \;\; PN \;\; V \;\; NP$. We say that a PCFG derivation is isomorphic to a STSG derivation if there is a corresponding PCFG subderivation for every step in the STSG derivation.

We will give a simple small PCFG with the following surprising property: for every subtree in the training corpus headed by A, the grammar will generate an isomorphic subderivation with probability $1/a$. In other words, rather than using the large, explicit STSG, we can use this small PCFG that generates isomorphic derivations, with identical probabilities.

The construction is as follows. For a node such as

$$\begin{array}{c} A@j \\ B@k \quad C@l \end{array}$$

we will generate the following eight PCFG rules, where the number in parentheses following a rule is its probability.

$$\begin{array}{llll} A_j \rightarrow BC & (1/a_j) & A \rightarrow BC & (1/a) \\ A_j \rightarrow B_k C & (b_k/a_j) & A \rightarrow B_k C & (b_k/a) \\ A_j \rightarrow BC_l & (c_l/a_j) & A \rightarrow BC_l & (c_l/a) \\ A_j \rightarrow B_k C_l & (b_k c_l/a_j) & A \rightarrow B_k C_l & (b_k c_l/a) \end{array} \tag{1}$$

We will show that subderivations headed by A with external non-terminals at the roots and leaves, internal non-terminals elsewhere have probability $1/a$. Subderivations headed by A_j with external non-terminals only at the leaves, internal non-terminals elsewhere, have probability $1/a_j$. The proof is by induction on the depth of the trees.

For trees of depth 1, there are two cases:

$$\begin{array}{c} A \\ B \quad C \end{array} \qquad \begin{array}{c} A@j \\ B \quad C \end{array}$$

Trivially, these trees have the required probabilities.

Now, assume that the theorem is true for trees of depth n or less. We show that it holds for trees of depth $n+1$. There are eight cases, one for each of the eight rules. We show two of them. Let $\begin{array}{c} B@k \\ \vdots \end{array}$ represent a tree of at most depth n with external leaves, headed by $B@k$, and with internal intermediate non-terminals. Then, for trees such as

$$\begin{array}{c} A@j \\ B@k \quad C@l \\ \vdots \qquad \vdots \end{array}$$

the probability of the tree is $\frac{1}{b_k}\frac{1}{c_l}\frac{b_k c_l}{a_j} = \frac{1}{a_j}$. Similarly, for another case, trees headed by

$$\begin{array}{c} A \\ B@k \quad C \end{array}$$

the probability of the tree is $\frac{1}{b_k}\frac{b_k}{a} = \frac{1}{a}$. The other six cases follow trivially with similar reasoning.

We call a PCFG derivation isomorphic to a STSG derivation if for every substitution in the STSG there is a corresponding subderivation in the PCFG. Figure 4 contains an example of isomorphic derivations, using two subtrees in the STSG and four productions in the PCFG.

We call a PCFG tree isomorphic to a STSG tree if they are identical when internal non-terminals are changed to external non-terminals. Our main theorem is that this construction produces PCFG trees isomorphic to the STSG trees with equal probability. If every subtree in the training corpus occurred exactly once, this would be trivial to prove. For every STSG subderivation, there would be an isomorphic PCFG sub-

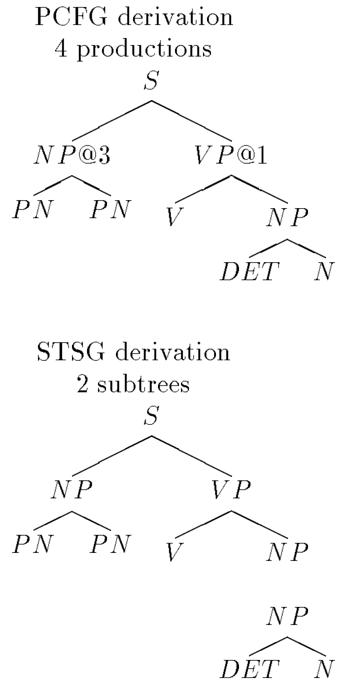
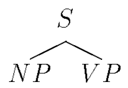


Figure 4: Example of Isomorphic Derivation

derivation, with equal probability. Thus for every STSG derivation, there would be an isomorphic PCFG derivation, with equal probability. Thus every STSG tree would be produced by the PCFG with equal probability.

However, it is extremely likely that some subtrees, especially trivial ones like



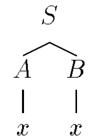
will occur repeatedly.

If the STSG formalism were modified slightly, so that trees could occur multiple times, then our relationship could be made one to one. Consider a modified form of the DOP model, in which when subtrees occurred multiple times in the training corpus, their counts were not merged: both identical trees are added to the grammar. Each of these trees will have a lower probability than if their counts were merged. This would change the probabilities of the derivations; however the probabilities of parse trees would not change, since there would be correspondingly more derivations for each tree. Now, the desired one to one relationship holds: for every derivation in the new STSG there is an isomorphic derivation in the PCFG with equal probability. Thus, summing over all derivations of a tree in the STSG yields the same probability as summing over all the isomorphic derivations in the PCFG. Thus, every STSG tree would be produced by the PCFG with equal probability.

It follows trivially from this that no extra trees are produced by the PCFG. Since the total probability of the trees produced by the STSG is 1, and the PCFG produces these trees with the same probability, no probability is "left over" for any other trees.

Parsing Algorithm

There are several different evaluation metrics one could use for finding the best parse. In the section covering previous research, we considered the most probable derivation and the most probable parse tree. There is one more metric we could consider. If our performance evaluation were based on the number of constituents correct, using measures similar to the crossing brackets measure, we would want the parse tree that was most likely to have the largest number of correct constituents. With this criterion and the example grammar of Figure 3, the best parse tree would be



The probability that the S constituent is correct is 1.0, while the probability that the A constituent is correct is $\frac{5}{9}$, and the probability that the B constituent is correct is $\frac{4}{9}$. Thus, this tree has on average 2 constituents correct. All other trees will have fewer constituents correct on average. We call the best parse tree under this criterion the *Maximum Constituents Parse*. Notice that this parse tree cannot even be produced by the grammar: each of its constituents is good, but it is not necessarily good when considered as a full tree.

Bod (1993a, 1995a) shows that the most probable derivation does not perform as well as the most probable parse for the DOP model, getting 65% exact match for the most probable derivation, versus 96% correct for the most probable parse. This is not surprising, since each parse tree can be derived by many different derivations; the most probable parse criterion takes all possible derivations into account. Similarly, the Maximum Constituents Parse is also derived from the sum of many different derivations. Furthermore, although the Maximum Constituents Parse should not do as well on the exact match criterion, it should perform even better on the percent constituents correct criterion. We have previously performed a detailed comparison between the most likely parse, and the Maximum Constituents Parse for Probabilistic Context Free Grammars (Goodman, 1996); we showed that the two have very similar performance on a broad range of measures, with at most a 10% difference in error rate (i.e., a change from 10% error rate to 9% error rate.) We therefore think that it is reasonable to use a Maximum Constituents Parser to parse the DOP model.

The parsing algorithm is a variation on the Inside-Outside algorithm, developed by Baker (1979) and discussed in detail by Lari and Young (1990). However, while the Inside-Outside algorithm is a grammar re-estimation algorithm, the algorithm presented here is just a parsing algorithm. It is closely related to a similar algorithm used for Hidden Markov Models (Rabiner, 1989) for finding the most likely state at each time. However, unlike in the HMM case where the algorithm produces a simple state sequence, in the PCFG case a parse tree is produced, resulting in addi-

```
for length := 2 to n
   for s := 1 to n-length+1
      t := s + length - 1;
      for all non-terminals X
         sum[X] := g(s, t, X);
      loop over addresses k
         let X := non-terminal at k;
         let sum[X] := sum[X] + g(s,t,X_k);
      loop over non-terminals X
         let max_X := arg max of sum[X]
      loop over r such that s <= r < t
         let best_split :=
            max of maxc[s,r] + maxc[r+1,t];
      maxc[s,t] := sum[max_X] + best_split;
```

Figure 5: Maximum Constituents Data-Oriented Parsing Algorithm

tional constraints.

A formal derivation of a very similar algorithm is given elsewhere (Goodman, 1996); only the intuition is given here. The algorithm can be summarized as follows. First, for each potential constituent, where a constituent is a non-terminal, a start position, and an end position, find the probability that that constituent is in the parse. After that, put the most likely constituents together to form a parse tree, using dynamic programming.

The probability that a potential constituent occurs in the correct parse tree, $P(X \stackrel{*}{\Rightarrow} w_s...w_t | S \stackrel{*}{\Rightarrow} w_1...w_n)$, will be called $g(s,t,X)$. In words, it is the probability that, given the sentence $w_1...w_n$, a symbol X generates $w_s...w_t$. We can compute this probability using elements of the Inside-Outside algorithm. First, compute the inside probabilities, $e(s,t,X) = P(X \stackrel{*}{\Rightarrow} w_s...w_t)$. Second, compute the outside probabilities, $f(s,t,X) = P(S \stackrel{*}{\Rightarrow} w_1...w_{s-1}Xw_{t+1}...w_n)$. Third, compute the matrix $g(s,t,X)$:

$$
\begin{aligned}
&g(s,t,X) \\
&\quad = \frac{P(S \stackrel{*}{\Rightarrow} w_1...w_{s-1}Xw_{t+1}...w_n)P(X \stackrel{*}{\Rightarrow} w_s...w_t)}{P(S \stackrel{*}{\Rightarrow} w_1...w_n)} \\
&\quad = f(s,t,X) \times e(s,t,X)/e(1,n,S)
\end{aligned}
$$

Once the matrix $g(s,t,X)$ is computed, a dynamic programming algorithm can be used to determine the best parse, in the sense of maximizing the number of constituents expected correct. Figure 5 shows pseudocode for a simplified form of this algorithm.

For a grammar with g nonterminals and training data of size T, the run time of the algorithm is $O(Tn^2 + gn^3 + n^3)$ since there are two layers of outer loops, each with run time at most n, and inner loops, over addresses (training data), nonterminals and n. However, this is dominated by the computation of the Inside and Outside probabilities, which takes time $O(rn^3)$, for a grammar with r rules. Since there are eight rules for every node in the training data, this is $O(Tn^3)$.

By modifying the algorithm slightly to record the actual split used at each node, we can recover the best parse. The entry `maxc[1, n]` contains the expected number of correct constituents, given the model.

Experimental Results and Discussion

We are grateful to Bod for supplying the data that he used for his experiments (Bod, 1995b, Bod, 1995a, Bod, 1993c). The original ATIS data from the Penn Tree Bank, version 0.5, is very noisy; it is difficult to even automatically read this data, due to inconsistencies between files. Researchers are thus left with the difficult decision as to how to clean the data. For this paper, we conducted two sets of experiments: one using a minimally cleaned set of data,[1] making our results comparable to previous results; the other using the ATIS data prepared by Bod, which contained much more significant revisions.

Ten data sets were constructed by randomly splitting minimally edited ATIS (Hemphill et al., 1990) sentences into a 700 sentence training set, and 88 sentence test set, then discarding sentences of length > 30. For each of the ten sets, both the DOP algorithm outlined here and the grammar induction experiment of Pereira and Schabes were run. Crossing brackets, zero crossing brackets, and the paired differences are presented in Table 1. All sentences output by the parser were made binary branching (see the section covering analysis of Bod's data), since otherwise the crossing brackets measures are meaningless (Magerman, 1994).

[1] A diff file between the original ATIS data and the cleaned up version, in a form usable by the "ed" program, is available by anonymous FTP from ftp://ftp.das.harvard.edu/pub/goodman/atis-ed/ ti_tb.par-ed and ti_tb.pos-ed. Note that the number of changes made was small. The diff files sum to 457 bytes, versus 269,339 bytes for the original files, or less than 0.2%.

Criteria	Min	Max	Range	Mean	StdDev
Cross Brack DOP	86.53%	96.06%	9.53%	90.15%	2.65%
Cross Brack P&S	86.99%	94.41%	7.42%	90.18%	2.59%
Cross Brack DOP−P&S	-3.79%	2.87%	6.66%	-0.03%	2.34%
Zero Cross Brack DOP	60.23%	75.86%	15.63%	66.11%	5.56%
Zero Cross Brack P&S	54.02%	78.16%	24.14%	63.94%	7.34%
Zero Cross Brack DOP−P&S	-5.68%	11.36%	17.05%	2.17%	5.57%

Table 1: DOP versus Pereira and Schabes on Minimally Edited ATIS

Criteria	Min	Max	Range	Mean	StdDev
Cross Brack DOP	95.63%	98.62%	2.99%	97.16%	0.93%
Cross Brack P&S	94.08%	97.87%	3.79%	96.11%	1.14%
Cross Brack DOP−P&S	-0.16%	3.03%	3.19%	1.05%	1.04%
Zero Cross Brack DOP	78.67%	90.67%	12.00%	86.13%	3.99%
Zero Cross Brack P&S	70.67%	88.00%	17.33%	79.20%	5.97%
Zero Cross Brack DOP−P&S	-1.33%	20.00%	21.33%	6.93%	5.65%
Exact Match DOP	58.67%	68.00%	9.33%	63.33%	3.22%

Table 2: DOP versus Pereira and Schabes on Bod's Data

A few sentences were not parsable; these were assigned right branching period high structure, a good heuristic (Brill, 1993).

We also ran experiments using Bod's data, 75 sentence test sets, and no limit on sentence length. However, while Bod provided us with his data, he did not provide us with the split into test and training that he used; as before we used ten random splits. The results are disappointing, as shown in Table 2. They are noticeably worse than those of Bod, and again very comparable to those of Pereira and Schabes. Whereas Bod reported 96% exact match, we got only 86% using the less restrictive zero crossing brackets criterion. It is not clear what exactly accounts for these differences.[2] It is also noteworthy that the results are much better on Bod's data than on the minimally edited data: crossing brackets rates of 96% and 97% on Bod's data versus 90% on minimally edited data. Thus it appears that part of Bod's extraordinary performance can be explained by the fact that his data is much cleaner than the data used by other researchers.

DOP does do slightly better on most measures. We performed a statistical analysis using a t-test on the paired differences between DOP and Pereira and Schabes performance on each run. On the minimally edited ATIS data, the differences were statistically insignificant, while on Bod's data the differences were statistically significant beyond the 98'th percentile. Our technique for finding statistical significance is more strenuous than most: we assume that since all test sentences were parsed with the same training data, all results of a single run are correlated. Thus we compare paired differences of entire runs, rather than of sentences or constituents. This makes it harder to achieve statistical significance.

Notice also the minimum and maximum columns of the "DOP−P&S" lines, constructed by finding for each of the paired runs the difference between the DOP and the Pereira and Schabes algorithms. Notice that the minimum is usually negative, and the maximum is usually positive, meaning that on some tests DOP did worse than Pereira and Schabes and on some it did better. It is important to run multiple tests, especially with small test sets like these, in order to avoid misleading results.

Timing Analysis

In this section, we examine the empirical runtime of our algorithm, and analyze Bod's. We also note that Bod's algorithm will probably be particularly inefficient on longer sentences.

It takes about 6 seconds per sentence to run our algorithm on an HP 9000/715, versus 3.5 hours

[2] Ideally, we would exactly reproduce these experiments using Bod's algorithm. Unfortunately, it was not possible to get a full specification of the algorithm.

to run Bod's algorithm on a Sparc 2 (Bod, 1995b). Factoring in that the HP is roughly four times faster than the Sparc, the new algorithm is about 500 times faster. Of course, some of this difference may be due to differences in implementation, so this estimate is fairly rough.

Furthermore, we believe Bod's analysis of his parsing algorithm is flawed. Letting G represent grammar size, and ϵ represent maximum estimation error, Bod correctly analyzes his runtime as $O(Gn^3\epsilon^{-2})$. However, Bod then neglects analysis of this ϵ^{-2} term, assuming that it is constant. Thus he concludes that his algorithm runs in polynomial time. However, for his algorithm to have some reasonable chance of finding the most probable parse, the number of times he must sample his data is at least inversely proportional to the conditional probability of that parse. For instance, if the maximum probability parse had probability 1/50, then he would need to sample at least 50 times to be reasonably sure of finding that parse.

Now, we note that the conditional probability of the most probable parse tree will in general decline exponentially with sentence length. We assume that the number of ambiguities in a sentence will increase linearly with sentence length; if a five word sentence has on average one ambiguity, then a ten word sentence will have two, etc. A linear increase in ambiguity will lead to an exponential decrease in probability of the most probable parse.

Since the probability of the most probable parse decreases exponentially in sentence length, the number of random samples needed to find this most probable parse increases exponentially in sentence length. Thus, when using the Monte Carlo algorithm, one is left with the uncomfortable choice of exponentially decreasing the probability of finding the most probable parse, or exponentially increasing the runtime.

We admit that this is a somewhat informal argument. Still, the Monte Carlo algorithm has never been tested on sentences longer than those in the ATIS corpus; there is good reason to believe the algorithm will not work as well on longer sentences. Note that our algorithm has true runtime $O(Tn^3)$, as shown previously.

Analysis of Bod's Data

In the DOP model, a sentence cannot be given an exactly correct parse unless all productions in the correct parse occur in the training set. Thus, we can get an upper bound on performance by examining the test corpus and finding which parse trees could not be generated using only productions in the training corpus. Unfortunately, while Bod provided us with his data, he did not specify which sentences were test and which were training. We can however find an upper bound on average case performance, as well as an upper bound on the probability that any particular level of performance could be achieved.

Bod randomly split his corpus into test and training. According to his thesis (Bod, 1995a, page 64), only one of his 75 test sentences had a correct parse which could not be generated from the training data. This turns out to be very surprising. An analysis of Bod's data shows that at least some of the difference in performance between his results and ours must be due to an extraordinarily fortuitous choice of test data. It would be very interesting to see how our algorithm performed on Bod's split into test and training, but he has not provided us with this split. Bod did examine versions of DOP that smoothed, allowing productions which did not occur in the training set; however his reference to coverage is with respect to a version which does no smoothing.

In order to perform our analysis, we must determine certain details of Bod's parser which affect the probability of having most sentences correctly parsable. When using a chart parser, as Bod did, three problematic cases must be handled: ϵ productions, unary productions, and n-ary ($n > 2$) productions. The first two kinds of productions can be handled with a probabilistic chart parser, but large and difficult matrix manipulations are required (Stolcke, 1993); these manipulations would be especially difficult given the size of Bod's grammar. Examining Bod's data, we find he removed ϵ productions. We also assume that Bod made the same choice we did and eliminated unary productions, given the difficulty of correctly parsing them. Bod himself does not know which technique he used for n-ary productions, since the chart parser he used was written by a third party (Bod, personal communication).

The n-ary productions can be parsed in a straightforward manner, by converting them to binary branching form; however, there are at least three different ways to convert them, as illustrated in Table 3. In method "Correct", the n-ary branching productions are converted in such a way that no overgeneration is introduced. A set of special non-terminals is added, one for each partial right hand side. In method "Continued", a single

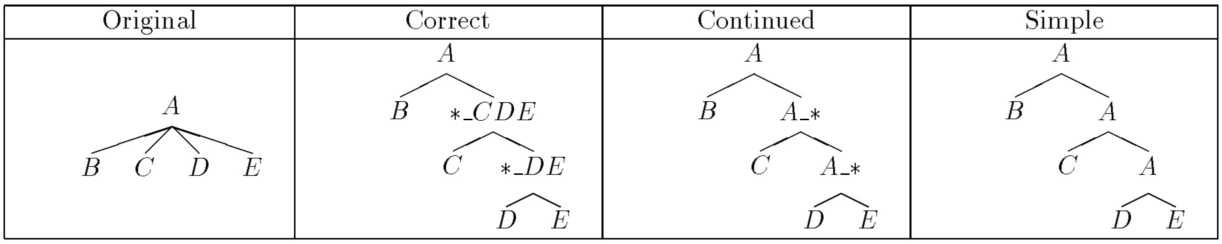


Table 3: Transformations from N-ary to Binary Branching Structures

	Correct		Continued		Simple	
no unary	0.78	0.0000002	0.88	0.0009484	0.90	0.0041096
unary	0.80	0.0000011	0.90	0.0037355	0.92	0.0150226

Table 4: Probabilities of Sentences with Unique Productions/Test Data with Ungeneratable Sentences

new non-terminal is introduced for each original non-terminal. Because these non-terminals occur in multiple contexts, some overgeneration is introduced. However, this overgeneration is constrained, so that elements that tend to occur only at the beginning, middle, or end of the right hand side of a production cannot occur somewhere else. If the "Simple" method is used, then no new non-terminals are introduced; using this method, it is not possible to recover the n-ary branching structure from the resulting parse tree, and significant overgeneration occurs.

Table 4 shows the undergeneration probabilities for each of these possible techniques for handling unary productions and n-ary productions.[3] The first number in each column is the probability that a sentence in the training data will have a production that occurs nowhere else. The second number is the probability that a test set of 75 sentences drawn from this database will have one ungeneratable sentence: $75p^{74}(1-p)$.[4]

The table is arranged from least generous to most generous: in the upper left hand corner is a technique Bod might reasonably have used; in that case, the probability of getting the test set he described is less than one in a million. In the lower right corner we give Bod the absolute maximum benefit of the doubt: we assume he used a parser capable of parsing unary branching productions, that he used a very overgenerating grammar, and that he used a loose definition of "Exact Match." Even in this case, there is only about a 1.5% chance of getting the test set Bod describes.

Conclusion

We have given efficient techniques for parsing the DOP model. These results are significant since the DOP model has perhaps the best reported parsing accuracy; previously the full DOP model had not been replicated due to the difficulty and computational complexity of the existing algorithms. We have also shown that previous results were partially due to an unlikely choice of test data, and partially due to the heavy cleaning of the data, which reduced the difficulty of the task.

Of course, this research raises as many questions as it answers. Were previous results due only to the choice of test data, or are the differences in implementation partly responsible? In that case, there is significant future work required to understand which differences account for Bod's exceptional performance. This will be complicated by the fact that sufficient details of Bod's implementation are not available.

This research also shows the importance of testing on more than one small test set, as well as the importance of not making cross-corpus comparisons; if a new corpus is required, then previous algorithms should be duplicated for comparison.

[3] A perl script for analyzing Bod's data is available by anonymous FTP from ftp://ftp.das.harvard.edu/pub/goodman/analyze.perl

[4] Actually, this is a slight overestimate for a few reasons, including the fact that the 75 sentences are drawn without replacement. Also, consider a sentence with a production that occurs only in one other sentence in the corpus; there is some probability that both sentences will end up in the test data, causing both to be ungeneratable.

References

[Baker, 1979] 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.

[Bod, 1992] Rens Bod. 1992. Mathematical properties of the data oriented parsing model. Paper presented at the *Third Meeting on Mathematics of Language (MOL3)*, Austin Texas.

[Bod, 1993a] Rens Bod. 1993a. Data-oriented parsing as a general framework for stochastic language processing. In K. Sikkel and A. Nijholt, editors, *Parsing Natural Language*. Twente, The Netherlands.

[Bod, 1993b] Rens Bod. 1993b. Monte Carlo parsing. In *Proceedings Third International Workshop on Parsing Technologies*, Tilburg/Durbury.

[Bod, 1993c] Rens Bod. 1993c. Using an annotated corpus as a stochastic grammar. In *Proceedings of the Sixth Conference of the European Chapter of the ACL*, pages 37–44.

[Bod, 1995a] Rens Bod. 1995a. *Enriching Linguistics with Statistics: Performance Models of Natural Language*. University of Amsterdam ILLC Dissertation Series 1995-14. Academische Pers, Amsterdam.

[Bod, 1995b] Rens Bod. 1995b. The problem of computing the most probable tree in data-oriented parsing and stochastic tree grammars. In *Proceedings of the Seventh Conference of the European Chapter of the ACL*.

[Brill, 1993] Eric Brill. 1993. *A Corpus-Based Approach to Language Learning*. Ph.D. thesis, University of Pennsylvania.

[Goodman, 1996] Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of the ACL*. To appear.

[Hemphill et al., 1990] Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *DARPA Speech and Natural Language Workshop*, Hidden Valley, Pennsylvania, June. Morgan Kaufmann.

[Lari and Young, 1990] 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.

[Magerman, 1994] David Magerman. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University University, February.

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

[Rabiner, 1989] L.R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), February.

[Scha, 1990] R. Scha. 1990. Language theory and language technology; competence and performance. In Q.A.M. de Kort and G.L.J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*. Landelijke Vereniging van Neerlandici (LVVN-jaarboek), Almere. In Dutch.

[Schabes et al., 1993] Yves Schabes, Michal Roth, and Randy Osborne. 1993. Parsing the Wall Street Journal with the Inside-Outside algorithm. In *Proceedings of the Sixth Conference of the European Chapter of the ACL*, pages 341–347.

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

[Sima'an, 1996] Khalil Sima'an. 1996. Efficient disambiguation by means of stochastic tree substitution grammars. In R. Mitkov and N. Nicolov, editors, *Recent Advances in NLP 1995*, volume 136 of *Current Issues in Linguistic Theory*. John Benjamins, Amsterdam.

[Stolcke, 1993] Andreas Stolcke. 1993. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. Technical Report TR-93-065, International Computer Science Institute, Berkeley, CA.