Monte Carlo Parsing

Rens Bod

Department of Computational Linguistics, University of Amsterdam
Spuistraat 134, NL-1012 VB AMSTERDAM
email: rens@alf.let.uva.nl

Abstract

In stochastic language processing, we are often interested in the most probable parse of an input string. Since there can be exponentially many parses, comparing all of them is not efficient. The Viterbi algorithm (Viterbi, 1967; Fujisaki et al., 1989) provides a tool to calculate in cubic time the most probable derivation of a string generated by a stochastic context free grammar. However, in stochastic language models that allow a parse tree to be generated by different derivations — like Data Oriented Parsing (DOP) or Stochastic Lexicalized Tree-Adjoining Grammar (SLTAG) — the most probable derivation does not necessarily produce the most probable parse. In such cases, a Viterbi-style optimisation does not seem feasible to calculate the most probable parse. In the present article we show that by incorporating Monte Carlo techniques into a polynomial time parsing algorithm, the maximum probability parse can be estimated as accurately as desired in polynomial time. Monte Carlo parsing is not only relevant to DOP or SLTAG, but also provides for stochastic CFGs an interesting alternative to Viterbi. Unlike the current versions of Viterbi-style optimisation (Fujisaki et al., 1989; Jelinek et al., 1990; Wright et al., 1991), Monte Carlo parsing is not restricted to CFGs in Chomsky Normal Form. For stochastic grammars that are parsable in cubic time, the time complexity of estimating the most probable parse with Monte Carlo turns out to be $O(n^3\varepsilon^{-2})$, where n is the length of the input string and ε the estimation error. In this paper we will treat Monte Carlo parsing first of all in the context of the DOP model, since it is especially here that the number of derivations generating a single tree becomes dramatically large. Finally, a Monte Carlo Chart parser is used to test the DOP model on a set of hand-parsed strings from the Air Travel Information System (ATIS) spoken language corpus. Preliminary experiments indicate 96% test set parsing accuracy.

1 Motivation

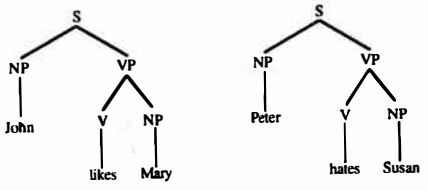
As soon as a formal grammar characterizes a non-trivial part of a natural language, almost every input string of reasonable length gets an unmanageably large number of different analyses. Since most of these analyses are not perceived as plausible by a human language user, there is a need for distinguishing the plausible parse(s) of an input string from the implausible ones. In stochastic language processing, it is assumed that the most plausible parse of an input string is its most probable parse. Most instantiations of this idea estimate the probability of a parse by assigning application probabilities to context free rewrite rules (Jelinek et al., 1990; Black et al., 1992; Briscoe — Carroll, 1993), or by assigning combination probabilities to elementary trees (Resnik, 1992; Schabes, 1992).

There is some agreement now that context free rewrite rules are not adequate for estimating the probability of a parse, since they do not capture lexical context, and hence do not describe how the probability of syntactic structures or lexical items depends on that context. In stochastic lexicalized tree-adjoining grammar (Schabes, 1992), this lack of context-sensitivity is overcome by assigning probabilities to larger structural units. However, it is not always evident which structures should be considered as elementary structures. In (Schabes, 1992), it is proposed to infer a stochas-

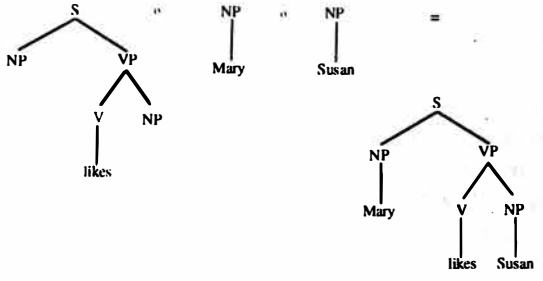
tic TAG from a large training corpus using an inside-outside-like iterative algorithm.

Data Oriented Parsing (DOP) (Scha, 1990,1992; Bod, 1992,1993), distinguishes itself from other statistical approaches in that it omits the step of inferring a grammar from a corpus. Instead, an annotated corpus is directly used as a stochastic grammar. An input string is parsed by combining subtrees from the corpus. In this view, every subtree can be considered as an elementary structure. As a consequence, one parse tree can usually be generated by several derivations that involve different subtrees. This leads to a statistics where the probability of a parse is equal to the sum of the probabilities of all its derivations. It is hoped that this approach can accommodate all statistical properties of a language corpus.

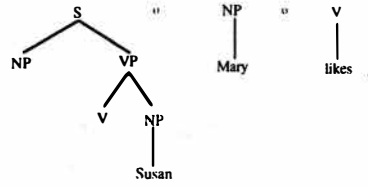
Let us illustrate DOP with an extremely simple example. Suppose that a corpus consists of only two trees:



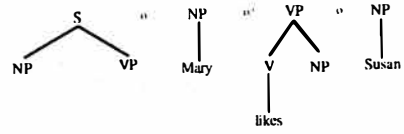
Suppose that our combination operation (indicated with ∘) consists of substituting a subtree on the leftmost identically labeled leaf node of another tree. Then the sentence *Mary likes Susan* can be parsed as an *S* by combining the following subtrees from the corpus. (For an exact definition of subtree, see section 2.)



But the same parse tree can also be derived by combining other corpus subtrees, for instance:



or



Thus, a parse can have several derivations involving different subtrees. These derivations have different probabilities. Using the corpus as our stochastic grammar, we estimate the probability of substituting a certain subtree on a specific node as the probability of selecting this subtree among all subtrees in the corpus that could be substituted on that node. The probability of a derivation can be computed as the product of the probabilities of the subtrees that are combined. As an example, we calculate the probability of the last derivation. The first subtree *S(NP,VP)* occurs twice in the corpus among a total of 20 subtrees rooted with an *S*. Thus, its probability is 2/20. The subtree *NP(Mary)* occurs once among a total of 4 subtrees that can be substituted on an *NP*, hence, its probability is 1/4. The probability of selecting the subtree *VP(V(likes),NP)* is 1/8, since there are 8 subtrees in the corpus rooted with a VP, among which this subtree occurs once. Finally, the probability of selecting *NP(Susan)* is equal to 1/4. The probability of the resulting derivation is then equal to 2/20 * 1/4 * 1/8 * 1/4 = 1/1280. The next table shows the probabilities of the three derivations given above.

P(1st example)	=	1/20 * 1/4 * 1/4	= 1/320
P(2nd example)	=	1/20 * 1/4 * 1/2	= 1/160
P(3rd example)	=	2/20 * 1/4 * 1/8 * 1/4	= 1/1280

This example illustrates that a statistical language model which defines probabilities over parses by taking into account only one derivation, does not accommodate all statistical properties of a language corpus. Instead, we define the probability of a parse as the sum of the probabilities of all its derivations. Finally, the probability of a string is equal to the sum of the probabilities of all its parses.

An important advantage of using a corpus for probability calculation, is that no training of parameters is needed, as is the case for other stochastic grammars (Jelinek et al., 1990; Pereira — Schabes, 1992; Schabes, 1992). Secondly, since we take into account all derivations of a parse, no relationship that might possibly be of statistical interest is ignored. Moreover, this approach does not suffer from a bias in favor of 'smaller' parse trees, as is the case with stochastic CFGs where derivations involving fewer rules, generating 'smaller' trees, are almost always favored regardless of the training material (Magerman — Marcus, 1991; Briscoe — Carroll, 1993). Finally, by using corpus subtrees directly as its structural units, DOP is largely independent of notation systems.

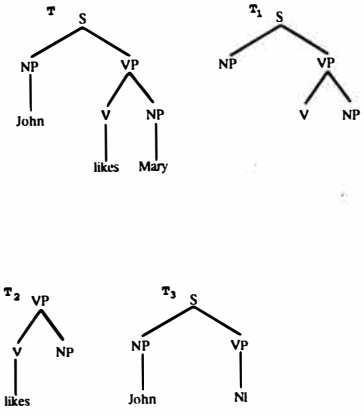
We will show that conventional parsing techniques can be applied to DOP. However, in order to find the most probable parse, a Viterbi-style algorithm does not seem feasible, since the most probable derivation does not necessarily produce the most probable parse. We will show that by using Monte Carlo techniques, the maximum probability parse can be estimated in polynomial time.

In the following, we first outline the DOP model in a more mathematical fashion, and provide an account of Monte Carlo parsing. Finally, we report on some experiments with a Monte Carlo Chart parser on the Air Travel Information System (ATIS) corpus as analyzed in the Penn Treebank.

2 The Data Oriented Parsing Model

A DOP model is characterized by a corpus of tree structures, together with a set of operations that combine subtrees from the corpus into new trees. In this section we explain more precisely what we mean by subtree, operations etc., in order to arrive at definitions of a parse and the probability of a parse with respect to a corpus.

A subtree of a tree T is a connected subgraph S of T such that for every node in S holds that if it has daughter nodes, then these are equal to the daughter nodes of the corresponding node in T. It is trivial to see that a subtree of a tree is also a tree. In the following example T_1 and T_2 are subtrees of T, whereas T_3 isn't.



The definition above also includes subtrees consisting of one node. Since such subtrees do not contribute to the parsing process, we exclude these pathological cases and consider only the set of subtrees consisting of more than one node. We shall use the following notation to indicate that a tree t is a subtree of a tree in a corpus C: $t \varepsilon C =_{\text{def}} \exists T \in C : t$ is a subtree of T, consisting of more than one node.

We will limit ourselves to the basic operation of *substitution*. (Other possible operations which combine subtrees are left to future research.) If t and u are trees, such that the *leftmost non-terminal leaf* of t is equal to the *root* of u, then $t \circ u$ is the tree that results from substituting this non-terminal leaf in t by tree u. The partial function $\circ$ is called *substitution*. We will write $(t \circ u) \circ v$ as $t \circ u \circ v$, and in general $(\ldots((t_1 \circ t_2) \circ t_3) \circ \ldots) \circ t_n$ as $t_1 \circ t_2 \circ t_3 \circ \ldots \circ t_n$.

Tree T is a *parse* of input string s with respect to a corpus C, iff the*yield* of T is equal to s and there are subtrees $t_1, \ldots, t_n \in C$, such that $T = t_1 \circ \ldots \circ t_n$. This definition correctly includes the trivial case of a subtree from the corpus whose yield is equal to the complete input string.

A *derivation* of a parse T with respect to a corpus C is a tuple of subtrees $< t_1, \ldots, t_n >$ such that $t_1, \ldots, t_n \varepsilon C$ and $t_1 \circ \ldots \circ t_n = T$.

Given a subtree $t_1 \varepsilon C$, a function root that yields the root of a tree, and a node labeled X, the conditional probability $P(t = t_1 \mid root(t) = X)$ denotes the probability that t_1 is substituted on X. If $root(t_1) = X$, this probability is 0. If $root(t_1) = X$, this probability can be estimated as the ratio between the number of occurrences of t_1 in C and the total number of occurrences of subtrees t' in C for which holds that $root\ (t\prime) = X$. Evidently, $\sum_i P(t = t_i \mid root(t) = X) = 1$ holds.

The probability of a derivation $< t1, \ldots, tn >$ is equal to the probability that the subtrees $t_1, \ldots, t_n$ are combined. This probability can be computed as the product of the conditional probabilities of the subtrees $t_1, \ldots, t_n$. Let $lnl(x)$ be the leftmost non-terminal leaf of tree x, then:

$$P(< t_1, \ldots, t_n >) = P(t = t_1 \mid root(t) = S) * \prod_{i=2 \text{ to } n} P(t = t_i \mid root(t) = lnl(t_1 \circ \ldots \circ t_{i-1}))$$

The probability of a parse is equal to the probability that any of its derivations occurs. Since the derivations are mutually exclusive, the probability of a parse is the sum of the probabilities of all its derivations. The conditional probability of a parse T given input string s, can be computed as the ratio between the probability of T and the sum of the probabilities of all parses of s.

The probability of a string is equal to the probability that any of its parses occurs. Since the parses are mutually exclusive, the probability of a string s can be computed as the sum of the probabilities of all its parses. It can be shown that $\sum_i P(s_i) = 1$ holds.

3 Monte Carlo Parsing

It is easy to show that in DOP, an input string can be parsed with conventional parsing techniques, by applying subtrees instead of rules to the string (Bod, 1992). Every subtree t can be seen as a production rule $root(t) \rightarrow t$, where the non-terminals of the yield of the right hand side constitute the symbols to which new rules/subtrees are applied. Given a cubic time parsing algorithm, the set of derivations of an input string, and hence the set of parses, can be calculated in cubic time. In order to select the most probable parse, it is not efficient to compare all parses, since there can be exponentially many of them. Although Viterbi's algorithm enables us to derive the most probable derivation in cubic time (Viterbi, 1967; Fujisaki et al., 1989; Wright et al., 1991), this algorithm does not seem feasible for DOP, since the most probable derivation does not necessarily produce the most probable parse. In DOP, a parse can be generated by exponentially many derivations. Thus, even for determining the probability of one parse, it is not efficient to add the probabilities of all derivations of that parse.

It is an open question, whether there exists an adaptation of the Viterbi algorithm that selects the maximum probability parse in cubic time for DOP. In this paper, we pursue an alternative approach. In order to estimate the maximum probability parse efficiently, we will apply Monte Carlo techniques to the decoding problem. We intend to show that, with Monte Carlo, the maximum probability parse can be estimated as accurately as desired, making its error arbitrarily small in polynomial time. Moreover, Monte Carlo techniques can easily be incorporated into virtually any polynomial time parsing algorithm. Thus, Monte Carlo parsing may also provide for stochastic CFGs an interesting alternative to Viterbi, which, in its current versions (Fujisaki et al., 1989; Jelinek et al., 1990; Wright et al., 1991), is restricted to CFGs in Chomsky Normal Form. We will treat Monte Carlo parsing first of all in the context of the DOP model, since it is especially here that the number of derivations generating a single tree

becomes dramatically large.

The essence of Monte Carlo is very simple: it estimates a probability distribution of events by taking random samples (Hammersley — Handscomb, 1964). The larger the samples we take, the higher the reliability. Since the events we are interested in are parses of a certain input string, we should randomly sample parses of that input string. The parse tree which is sampled most often is an estimation of the maximum probability parse. We can estimate the maximum probability parse as accurately as we want by choosing the number of randomly sampled parses as large as we want. The probability of a certain parse T given input string s can be estimated by dividing the number of occurrences of T by the total number of sampled parses N. According to the (Strong) Law of Large Numbers, the estimated probability converges to the actual probability. In the limit of N going to infinity, the estimated probability equals the actual probability: $P(T \mid s) = \#T/N$. From a classical result of probability theory (Chebyshev's inequality) it follows that, independently of the distribution, the time complexity of achieving a maximum estimation error ε by means of random sampling, is equal to $O(\varepsilon^{-2})$.

Let us now turn to the question of how to randomly sample a number of parses of an input string. The most straightforward way seems to be the following: first the set of parses of an input string is derived, yielding a shared parse forest. Next, random samples are taken from this forest, by randomly retrieving parses. Starting for instance at the S-node, a random expansion from the possible expansions is chosen at every node, taking into account the relative frequencies. The parse which is sampled most often is an estimation of the maximum probability parse. Given a cubic time parsing algorithm and assuming that the construction of a parse forest and the retrieval and comparing of parses can be done in cubic time (Leermakers, 1991), the time complexity of this method is $O(n^3\varepsilon^{-2})$ for a string of length n and an estimation error ε.

Depending on the size and the redundancy of the corpus, this method is not always the most efficient one. Instead of applying Monte Carlo techniques after the parsing process, we might also incorporate them *into* the parsing process. This second method consists of calculating a random subset of the parses. Instead of taking into account all candidates[1] at every node in the parsing process, we take a random sample from the total number of candidates at every node. In this way, a set of parses is calculated which is smaller than the total set of parses of an input string. Repeating this process allows us to randomly generate as many parses of a string as desired. If no parses are found during a round, the samples from the candidates may be increased until at least one parse is generated. If, instead, for a new input string a large number of parses is found, the current value of the sample size may be decreased again, and so forth. In the worst case the sample size equals 100% of the total number of candidates and no speedup is achieved. However, this can only happen with non-ambiguous grammars where every string has exactly one derivation. For an ambiguous grammar, any ambiguous string can always be parsed by taking samples from the candidates smaller than the total number of candidates (except that taking a sample from 1 candidate must yield at least that candidate). In our experiments with the ATIS corpus (see next section), it turned out that taking maximally 5% of the candidate subtrees, sufficed to calculate at least one parse for the input string (though often more were found).

As to the time complexity of this second method, it might seem that calculating a subset of exponentially many parses, will yield again exponentially many parses. And comparing exponentially many parses takes exponential time. Nevertheless, by taking the sample sizes relatively small, a tractable upper bound N can be defined, which, if exceeded by the number of parses generated sofar, serves as a stop condition in the repeated parsing process. Secondly, N can be made arbitrarily large, in order to make the estimation error ε arbitrarily small in, as we have seen, quadratic time. Hence, given a cubic time parsing algorithm and assuming that the sample sizes can be made smaller than the total number of candidates but large enough to generate at least one parse (as is the case for redundant grammars like DOP), the time complexity of this method is $O(n^3\varepsilon^{-2})$. Often it suffices to stop repeating the algorithm if the total number of parses ex-

[1] I.e. 'predictions' or 'proposed edges', depending of the kind of parser used.

ceeds a pre-determined bound N. The most frequently generated parse is then an estimation of the maximum probability parse. We shall see in the next section that for the ATIS corpus it sufficed to limit the number of randomly calculated parses to 100, in order to get high parsing accuracy. Though such a small sample may yield inaccurate probabilities for the single parses, it apparently suffices to determine which parse is the most probable one.

Although the worst time complexity of this second method is equivalent to that of the first one, the actual time cost turns out to be much lower. This can be explained by the fact that in the second method only a small part of the actual grammar is used. Since arbitrary CFGs are parsable in $| G |^2$ time, parsing a string 100 times using 5% of the grammar tends to be more efficient than parsing the same string only once using the whole grammar. Secondly, it turns out that the probability estimation of the second method also converges significantly faster. Thus, it seems that this method is especially apt to stochastic parsing with huge amounts of redundant data.

It should be stressed that incorporating Monte Carlo techniques into a parsing algorithm is only feasible if the samples from the candidates can be made much smaller than the total number of candidates, but still large enough to generate at least one parse. Secondly, the demanded maximum error should not be too small, in order to keep the actual time cost to an acceptable degree. For those interested in the Theory of Computation: the algorithms which employ the Monte Carlo techniques described here, are probabilistic algorithms belonging to the class of **B**ounded error **P**robabilistic **P**olynomial time (**BPP**) algorithms. BPP-problems are characterized as follows: it may take exponential time to solve them exactly, but there exists an estimation algorithm with a probability of error that becomes arbitrarily small in polynomial time.

4 Experiments

In order to test the DOP-model, in principle any annotated corpus can be used. This is one of the advantages of DOP: its independence of a notation system. For our experiments[2], we used the naturally occurring Air Travel Information System (ATIS) corpus (Hemphill et al., 1990) as analyzed in the Pennsylvania Treebank (Marcus, 1991; Santorini, 1991). This corpus is of interest since it is used by the DARPA community to evaluate their grammars and speech systems.

We used the standard method of randomly dividing the corpus into a 90% training set and a 10% test set. The 675 trees from the training set were directly used as our stochastic grammar, from which the subtrees and their relative frequencies were derived. The 75 part-of-speech sequences from the test set served as input strings that were parsed with the training set using a Monte Carlo Chart parser (Mijnlief, 1993). To establish the performance of the system, the parsing results were then compared with the trees in the test set. (Note that the "correct" parse was decided beforehand, and not afterwards.)

To measure accuracy, one often uses the notion of *bracketing* accuracy, i.e. the percentage of brackets of the analyses that are not "crossing" the bracketings in the Treebank (Black et al., 1991; Harrison et al., 1991; Pereira — Schabes, 1992; Grishman et al., 1992; Schabes et al., 1993). We believe, however, that the notion of bracketing accuracy is too poor for measuring the performance of a parser. A test set can have a high bracketing accuracy, whereas the percentage of sentences in which no crossing bracket is found (*sentence* accuracy) is extremely low. In (Schabes et al., 1993), it is shown that for sentences of 10 to 20 words (taken from the Wall Street Journal corpus), a bracketing accuracy of 82.5% corresponds to a sentence accuracy of 30%, whereas for sentences of 20 to 30 words a bracketing accuracy of 71.5% corresponds to a sentence accuracy of 6.8%! We shall employ the even stronger notion of *parsing* accuracy, defined as the percentage of the test sentences for which the maximum probability parse is *identical* to the test set parse in the Treebank.

It is one of the most essential features of the DOP approach, that arbitrarily large subtrees are taken into consideration. In order to test the usefulness of this feature, we performed different experiments constraining the *depth* of the subtrees. The depth of a tree is defined as the length of its longest path. The following table shows the results of seven experiments. The accuracy refers to

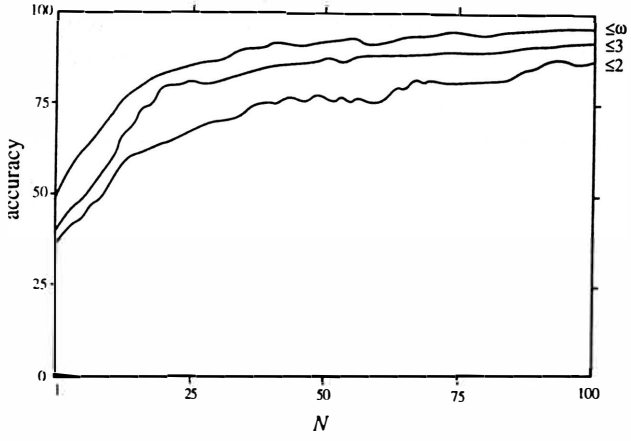
[2]Some of the experiments reported were published in (Bod, 1993).

the parsing accuracy at $N = 100$ sampled parses, and is rounded off to the nearest integer.

depth	accuracy
≤ 2	87%
≤ 3	92%
≤ 4	93%
≤ 5	93%
≤ 6	95%
≤ 7	95%
unbounded	96%

Parsing accuracy for the ATIS corpus, at $N = 100$

The table shows that there is a relatively rapid increase in parsing accuracy when enlarging the maximum depth of the subtrees to 3. The accuracy keeps increasing, at a slower rate, when the depth is enlarged further. The highest accuracy is obtained by using all subtrees from the corpus: 72 out of the 75 sentences from the test set are parsed correctly. In the following figure, parsing accuracy is plotted against the number of randomly generated parses N for three of our experiments: the experiments where the depth of the subtrees is constrained to 2 and 3, and the experiment where the depth is unconstrained.



Parsing accuracy for the ATIS corpus, with depth ≤ 2, with depth ≤ 3 and with unbounded depth

It might also be interesting to look in detail at some parses derived with different constraints on the depths of the subtrees. Consider the test sentence "Arrange the flight code of the flight from Denver to Dallas Worth in descending order", which corresponds to the p-o-s sequence "* VB DT NN NN IN DT NN IN NP TO NP NP IN VBG NN".[3] According to the Treebank, this sentence has the following structure (for a description of the notation system see (Santorini, 1990,1991)):

```
S NP *
  VP VB Arrange
     NP NP DT the
           NN flight
           NN code
        PP IN of
           NP NP DT the
                 NN flight
              PP PP IN from
                    NP NP Denver
                 PP TO to
                    NP NP Dallas
                       NP Worth
     PP IN in
        NP VP VBG descending
           NN order
```

Limiting the depth of the subtrees to 2, the following maximum probability parse was estimated for this string (where for reasons of readability the lexical items are added to the p-o-s tags):

```
S NP *
  VP VB Arrange
     NP NP DT the
           NN flight
           NN code
        PP IN of
           NP NP DT the
                 NN flight
              PP PP IN from
                    NP NP Denver
                 PP TO to
                    NP NP Dallas
                       NP Worth
              PP IN in
                 NP VP VBG descending
                    NN order
```

In this parse tree, we see that the prepositional phrase "in descending order" is incorrectly

[3]Empty elements, like *, had to be treated as part-of-speech elements, in order to be able to use the training set directly as a grammar.

attached to the NP "the flight" instead of to the verb "arrange". This false attachment might be explained by the high relative frequencies of the following subtrees with depth 2 (that appear in structures of sentences like "Show me the transportation from SFO to downtown San Francisco in August", where the PP "in August" is attached to the NP "the transportation", and not to the verb "show").

```
NP NP          NP NP
   PP             PP PP
   PP IN             PP
      NP          PP IN
                     NP
```

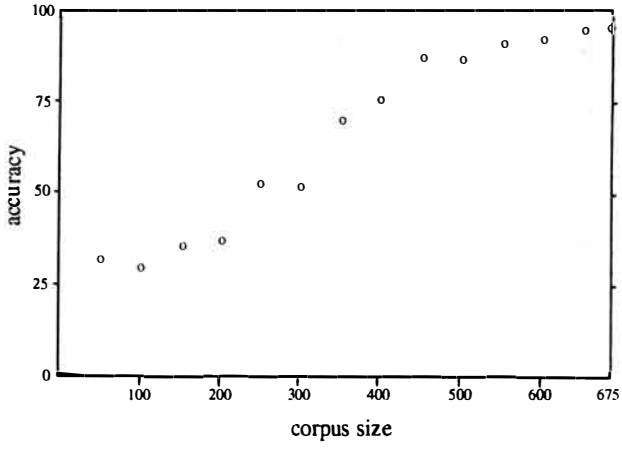
Only if the maximum depth of the subtrees was enlarged to 4, subtrees like the following could be sampled, which led to the estimation of the correct parse tree.

```
VP VB
   NP NP
      PP
   PP IN
      NP VP VBG
         NN
```

It is interesting to note that this subtree occurs only once in the corpus. Nevertheless, it induces the correct parsing of the test sentence. This seems to contradict the observation that probabilities based on sparse data are not reliable (Gale — Church, 1990; Magerman — Marcus, 1991). Since many large subtrees are once-occurring events (hapaxes), there seems to be a preference in DOP for an occurence-based approach if enough context is provided: large subtrees, even if they occur once, tend to contribute to the generation of the correct parse, since they provide much contextual information. Although these subtrees have low probabilities, they tend to induce the correct parse because fewer subtrees are needed to construct a derivation, and therefore the probability of such a derivation tends to be higher than a derivation constructed by many small highly frequent subtrees.

Additional experiments seemed to confirm this hypothesis. Throwing away all hapaxes, yielded an accuracy of 92% (without constraints on the depth of the subtrees and for N = 100), which is a decrease of 4%. Distinguishing between small and large hapaxes, showed that the accuracy was not affected by filtering the subtrees from hapaxes smaller than depth 2 (although the convergence seemed to be slightly faster). Eliminating the hapaxes larger than depth 3, however, decreased the accuracy. Thus, statistical reliability seems only to be relevant if not enough contextual information is available. In such a case, best guesses must be as reliable as possible. When much structural/contextual information is known, on the other hand, there tends to be only one choice. This seems to correspond to the fact that small parts of sentences tend to have many more real structural ambiguities (since not enough information is known) than longer subsentences or whole sentences.

Given the high accuracy achieved by the experiments, we might conclude that the ATIS corpus is a relatively large corpus for its small domain, where almost all relevant constructions occur. It seemed interesting to know how much the accuracy depends on the size of the corpus. For studying this question, we performed additional experiments with different corpus sizes. Starting with a corpus of only 50 parse trees (randomly chosen from the initial training corpus of 675 trees), we increased its size with intervals of 50. As our test set, we took the same 75 p-o-s sequences as used in the previous experiments. In the next figure the parsing accuracy, for $N = 100$, is plotted against the corpus size, using all corpus subtrees.



Parsing accuracy for the ATIS corpus, with unbounded depth.

The figure shows the increase in parsing accuracy. For a corpus size of 450 trees, the accuracy reaches already 88%. After this, the growth decreases, but the accuracy is still growing at corpus size 675. Thus, we might expect an even higher accuracy if the corpus is further enlarged.

Finally, it might be interesting to compare our results with those of others. In (Pereira — Schabes, 1992), 90.36% *bracketing* accuracy was reported using a stochastic CFG trained on bracketings from the ATIS corpus. As said above, the notion of bracketing accuracy is much poorer than that of parsing accuracy. Thus, our pilot experiment suggests that our model has better performance than a stochastic CFG. Some work that achieved high *parsing* accuracy, though with different test data, are the parsers **P**earl and **P**icky of (Magerman — Marcus, 1991) and (Magerman — Weir, 1992). In their work, a stochastic CFG is combined with trigram statistics, yielding about 90% parsing accuracy with word sequences as input strings. We do not yet know what accuracy is achieved if DOP is directly tested on word sequences, instead of on p-o-s sequences. It is likely, that larger corpora are needed for this task.

5 Conclusions

Although a Viterbi-style algorithm provides a tool to derive in cubic time the most probable derivation generated by a stochastic context free grammar, this algorithm does not seem feasible for stochastic language models that allow a parse tree to be generated by different derivations (like DOP or SLTAG), since the most probable derivation does not necessarily produce the most probable parse.

We showed that, by incorporating Monte Carlo techniques into a polynomial parsing algorithm, the most probable parse can be estimated as accurately as desired, making its error arbitrarily small in polynomial time. For stochastic grammars that are parsable in cubic time, the time complexity of estimating the most probable parse with Monte Carlo turns out to be $O(n^3\varepsilon^{-2})$, for a string of length n and an estimation error ε. We suggested that Monte Carlo parsing may also provide for stochastic CFGs an interesting alternative to Viterbi, which, in its current versions, is restricted to CFGs in Chomsky Normal Form. Nevertheless, Monte Carlo parsing seems especially apt to stochastic parsing with huge amounts of redundant data, where one parse is generated by exponentially many (different) derivations.

A Monte Carlo Chart parser was used to test the DOP model on a set of hand-parsed strings from the ATIS corpus. It sufficed to limit the number of randomly calculated parses to 100, in order to get satisfying convergence with high parsing accuracy. It turned out that parsing accuracy improved if larger subtrees were used. Our experiments suggest that statistical reliability is only relevant if not enough structural/contextual information is available.

Acknowledgements

We thank Remko Scha for valuable comments on an earlier version of this paper, and Mitch Marcus for supplying the ATIS corpus.

References

Black E. et al. (1991) "A Procedure for Quantitatively Comparing the Syntactic Coverage of English". In: *Proceedings DARPA Speech and Natural Language Workshop*, Pacific Grove, Morgan Kaufmann.

Black E. — J. Lafferty — S. Roukos (1992) "Development and Evaluation of a Broad-Coverage Probabilistic Grammar of English-Language Computer Manuals". In: *Proceedings ACL'92*, Newark, Delaware.

Bod, R. (1992) "A Computational Model of Language Performance: Data Oriented Parsing". In: *Proceedings COLING'92*, Nantes.

Bod, R. (1993) "Using an Annotated Corpus as a Stochastic Grammar". In: *Proceedings EACL'93*, Utrecht.

Briscoe, T. — J. Carroll (1993) "Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars". In: *Computational Linguistics* 19(1), 25–59.

Fujisaki, T. — F. Jelinek — J. Cocke — E. Black — T. Nishino (1989) "A Probabilistic Method for Sentence Disambiguation". In: *Proceedings 1st Int. Workshop on Parsing Technologies*, Pittsburgh.

Gale, W. — K. Church (1990) "Poor Estimates of Context are Worse than None". In: *Proceedings DARPA Speech and Natural Language Workshop*, Hidden Valley, Morgan Kaufmann.

Grishman, R. — C. Macleod — J. Sterling (1992) "Evaluating Parsing Strategies Using Standardized Parse Files". In: *Proceedings ANLP'92*, Trento.

Hammersley, J. M. — D.C. Handscomb (1964) *Monte Carlo Methods*, Chapman and Hall, London.

Harrison, P. et al. (1991) "Evaluating Syntax Performance of Parser/Grammars". In: *Proceedings of the Natural Language Processing Systems Evaluation Workshop*, Berkeley.

Hemphill C. T. — J.J. Godfrey — G.R. Doddington (1990) "The ATIS spoken language systems pilot corpus". In: *Proceedings DARPA Speech and Natural Language Workshop*, Hidden Valley, Morgan Kaufmann.

Jelinek, F. — J.D. Lafferty — R.L. Mercer (1990) *Basic Methods of Probabilistic Context Free Grammars*, Technical Report IBM RC 16374 (#72684), Yorktown Heights.

Leermakers, R. (1991) "Non-deterministic Recursive Ascent Parsing". In: *Proceedings EACL'91*, Berlin.

Magerman, D. — M. Marcus (1991) "Pearl: A Probabilistic Chart Parser". In: *Proceedings EACL'91*, Berlin.

Magerman, D.— C. Weir (1992) "Efficiency, Robustness and Accuracy in Picky Chart Parsing". In: *Proceedings ACL'92*, Newark, Delaware.

Marcus, M. (1991) "Very Large Annotated Database of American English". *DARPA Speech and Natural Language Workshop*, Pacific Grove, Morgan Kaufmann.

Mijnlief, A. (1993) *A Monte Carlo Chart Parser for DOP*, Dept. of Computational Linguistics, University of Amsterdam.

Pereira, F. — Y. Schabes (1992) "Inside-Outside Reestimation from Partially Bracketed Corpora". In: *Proceedings ACL'92*, Newark.

Resnik, P. (1992) "Probabilistic Tree-Adjoining Grammar as a Framework for Statistical Natural Language Processing". In: *Proceedings COLING'92*, Nantes.

Santorini, B. (1990) *Part-of-Speech Tagging Guidelines for the Penn Treebank Project*, Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia.

Santorini, B. (1991) *Bracketing Guidelines for the Penn Treebank Project*, Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia.

Scha, R. (1990) "Language Theory and Language Technology; Competence and Performance" (in Dutch). In Q.A.M. de Kort & G.L.J. Leerdam (eds.), *Computertoepassingen in de Neerlandistiek*, Almere: Landelijke Vereniging van Neerlandici (LVVN-jaarboek).

Scha, R. (1992) "Virtual Grammars and Creative Algorithms" (in Dutch), *Gramma/TTT* 1(1).

Schabes, Y. (1992) "Stochastic Lexicalized Tree-Adjoining Grammars". In: *Proceedings COLING'92*, Nantes.

Schabes, Y. — M. Roth — R. Osborne (1993) "Parsing the Wall Street Journal with the Inside-Outside Algorithm". In: *Proceedings EACL'93*, Utrecht.

Viterbi, A. (1967) "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Trans. Information Theory*, IT-13, 260-269.

Wright, J. — E. Wrigley — R. Sharman (1991) "Adaptive Probabilistic Generalized LR Parsing". In: *Proceedings 2nd Int. Workshop on Parsing Technologies*, Cancun, Mexico.