Parsimonious Data-Oriented Parsing

Willem Zuidema
Institute for Logic, Language and
Computation, University of Amsterdam
Plantage Muidergracht 24
1018 TV, Amsterdam, the Netherlands
jzuidema@science.uva.nl

Abstract

This paper explores a parsimonious approach to Data-Oriented Parsing. While allowing, in principle, all possible subtrees of trees in the treebank to be productive elements, our approach aims at finding a manageable subset of these trees that can accurately describe empirical distributions over phrase-structure trees. The proposed algorithm leads to computationally much more tracktable parsers, as well as linguistically more informative grammars. The parser is evaluated on the OVIS and WSJ corpora, and shows improvements on efficiency, parse accuracy and testset likelihood.

1 Data-Oriented Parsing

Data-Oriented Parsing (DOP) is a framework for statistical parsing and language modeling originally proposed by Scha (1990). Some of its innovations, although radical at the time, are now widely accepted: the use of fragments from the trees in an annotated corpus as the symbolic grammar (now known as "treebank grammars", Charniak, 1996) and inclusion of all statistical dependencies between nodes in the trees for disambiguation (the "all-subtrees approach", Collins & Duffy, 2002).

The best known instantiations of the DOP-framework are due to Bod (1998; 2001; 2003), using the Probabilistic Tree Substitution Grammar (PTSG) formalism. Bod has advocated a *maximalist* approach to DOP, inducing grammars that contain all subtrees of all parse trees in the treebank, and using them to parse unknown sentences where all of these subtrees can potentially contribute to the most probable parse. Although Bod's empirical results have been excellent, his maximalism poses important computational challenges that, although not necessarily unsolvable, threaten both the scalability to larger treebanks and the cognitive plausibility of the models.

In this paper I explore a different approach to DOP, that I will call "Parsimonious Data-Oriented Parsing" (P-DOP). This approach remains true to Scha's original program, by allowing, in principle, all possible subtrees of trees in the treebank to be the productive elements. But unlike Bod's approach, P-DOP aims at finding a succinct subset of such elementary trees, chosen such that it can still accurately describe observed distributions over phrase-structure trees. I will demonstrate that P-DOP leads to computationally more tracktable parsers, as well as linguistically more informative grammars. Moreover, as P-DOP is formulated as an enrichment of the treebank Probabilistic Context-free Grammar (PCFG), it allows for much easier comparison to alternative approaches to statistical parsing (Collins, 1997; Charniak, 1997; Johnson, 1998; Klein and Manning, 2003; Petrov et al., 2006).

2 Independence Assumptions in PCFGs

Parsing with treebank PCFGs, in its simplest form, involves the following steps: (1) a treebank is created by extracting phrase-structure trees from an annotated corpus, and split in a train- and a testset; (2) a PCFG is read off from all productions in the trainset trees, with weights proportional to their fre-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 551–560, Prague, June 2007. ©2007 Association for Computational Linguistics

quency in the treebank (the "relative frequency estimate"); (3) a standard PCFG parser is used to find for each yield of the test-set trees the most probable parse; (4) these parses are compared to the test-set trees to count matching brackets, labels and trees.

PCFGs incorporate a strong statistical independence assumption: that the expansion of a nonterminal node is only dependent on the node's label. All state-of-the-art wide-coverage parsers relax this assumption in some way, for instance by (i) changing the parser in step (3), such that the application of rules is *conditioned* on other steps in the derivation process (Collins, 1997; Charniak, 1997), or by (ii) *enriching* the nonterminal labels in step (1) with context-information (Johnson, 1998; Klein and Manning, 2003), along with suitable backtransforms in step (4). These two approaches often turn out to be equivalent, although for some conditionings it is not trivial to work out the equivalent enrichment and vice versa, especially when combined with smoothing. Interesting recent work has focused on the automatic induction of enrichments (Matzuzaki et al., 2005; Prescher, 2005), leading to extremely accurate parsers (Petrov et al., 2006).

DOP relaxes the independence assumption by changing the class of probabilistic grammars induced in step (2). In DOP1 (Bod, 1998), a PTSG is induced, which consists, subject to some heuristic constraints, of all subtrees[1] of the treebank trees with a weight proportional to their frequency. PTSGs allow multiple derivations to yield the same parse; in DOP1 the sum of their probabilities gives the probability of the parse. The relation between DOP and enrichment/conditioning models was clarified by Joshua Goodman, who devised an efficient PCFG transform of the DOP1 model (Goodman, 1996). The size of the PCFG resulting from this transform is linear in the number of nonterminals tokens in the corpus. Goodman's transform, in combination with a range of heuristics, allowed Bod (2003) to run the DOP model on the Penn Treebank WSJ benchmark and obtain some of the best results obtained with a generative model.

The computational challenges for DOP are far from solved, however. The difference with style (ii) enrichment is that we derive many more rules from every original tree than the number of CFG-productions it contains. This is one reason why the relative frequency estimator for DOP is inconsistent (Johnson, 2002). But worse, perhaps, the size of the grammar remains gigantic[2], making it difficult for many in the field to replicate Bod's results.

In this paper, we develop a parsimonious approach to DOP, that avoids many of the computational problems of the maximalist approach but tries to maintain its excellent empirical performance. Our approach starts, both conceptually and technically, with an analysis of where the PCFG independence assumption breaks down when modeling empirical distributions. In section 2 we derive equations for the expected frequency of arbitrary subtrees under a distribution defined by a given PCFG, and use them to measure how much observed subtree-frequencies deviate from expectation. In section 4 we generalize this analysis to PTSGs. In section 5 we discuss an algorithm for estimating PTSGs from a treebank, that is based on minimizing the differences between expected and observed subtree-frequencies. We then proceed with discussing PTSGs induced from various treebanks, and in section 6 the use of these PTSGs for parsing.

3 Deviations from a PCFG distribution

PCFGs can be viewed as PTSGs where the elementary trees are restricted to depth 1; we therefore start by repeating the definition of PTSGs (Bod, 1998), and use notation appropriate for PTSGs throughout. An PTSG is a 5-tuple $\langle V_n, V_t, S, T, w\rangle$, where V_n is the set of non-terminal symbols; V_t is the set of terminal symbols; $S \in V_n$ is the start symbol; T is a set of elementary trees, such that for every $\tau \in T$ the unique root node $r(\tau) \in V_n$, the (possibly empty) set of internal nodes $i(\tau) \subset V_n$ and the set of leaf nodes $l(\tau) \subset V_n \cup V_t$; finally, $w : T \to [0, 1]$ is a probability (weight) distribution over the elementary trees, such that for any $\tau \in T$, $\sum_{\tau' \in R(\tau)} w(\tau') = 1$, where $R(\tau)$ is the set of elementary trees with the same root label as τ. It will prove useful to also define the set of all possible trees θ over the defined

[1] A subtree t' of a parse tree t is a tree such that every node i' in t' equals a node i in t, and i' either has no daughters or the same daughter nodes as i.

[2] Sections 2-21 of WSJ contain 1676821 productions. Of these,10^6 are lexical productions, and 36151 top-productions, leaving approx. 640000 internal productions which yield about 2.5×10^6 rules in Goodman's transform.

alphabets (with the same conditions on root, internal and leaf nodes as for T), and the set of all possible *complete* parse trees Θ (with $r(t) = S$ and all leaf nodes $l(t) \subset V_t$). Obviously, $T \subset \theta$ and $\Theta \subset \theta$.

The substitution operation $\circ$ is defined if the *leftmost nonterminal leaf* in τ_1 is identical to the root of τ_2. Performing substitution $\tau_1 \circ \tau_2$ yields t_3, if t_3 is identical to τ_1 with the leftmost nonterminal leaf replaced by τ_2. A derivation is a sequence of elementary trees, where the first tree $\tau \in T$ has root-label S and every next tree combines through substitution with the result of the substitutions before it. In this paper, we are only concerned with grammars that define proper probability distributions over trees, such that the probability of all derivations sum up to 1 and no probability mass gets lost in derivations that never reach a terminal yield. That is, we require (if $t(d)$ is the tree derived by derivation d):

$$\sum_{d:t(d)\in\Theta} P(d) = 1. \tag{1}$$

For simplicity, but without loss of generality, we assume there are no recursions on the start symbol.

In this section, we restrict ourselves to PCFG distributions, and thus to a T with only depth 1 trees. The probability of a PCFG rule (conditioned on its left-hand side) in the conventional notation, $P(A \mapsto \alpha\beta\ldots\gamma|A)$, now corresponds to the probability of a depth 1 tree (conditioned on its root nonterminal):

$$P\left(\begin{array}{c} A \\ \alpha \quad \beta \quad \ldots \quad \gamma \end{array} |A\right)$$

Of course, the probability of a (complete) derivation is simply the product of the (conditional) probabilities of the rules in the derivation. It is useful to consider, for a given grammar G generating a corpus of N trees, the expected frequency of visiting nonterminal state X:

$$\mathcal{E}F(X) = \begin{cases} N & \text{if } X = S \\ \sum_\tau \mathcal{E}F(\tau)C(X, l(\tau)) & \text{otherwise} \end{cases} \tag{2}$$

where $C(X, l(\tau))$ gives the number of occurrences of nonterminal X among the leaves of elementary tree τ. Furthermore, the expected *usage frequency* of τ is given by

$$\begin{aligned} \mathcal{E}F(\tau) &= \mathcal{E}F(r(\tau))P(\tau|r(\tau)) \\ &= \mathcal{E}F(r(\tau))w(\tau) \end{aligned} \tag{3}$$

Substituting eq (3) into (2) yields a system of $|V_n|$ linear equations, that can be straightforwardly solved using standard methods.

We are interested in the empirical deviations from the distribution defined by a given grammar (for instance, the treebank PCFG), such that we can adjust the grammar to better model the training data (whilst avoiding overfitting). In line with the general DOP approach, we would like to measure this deviation for every possible subtree. Of course, the conditional probability of an arbitrary subtree is simply the product of the rule probabilities. The expected frequency of a subtree is the expected frequency of its root state, times the conditional probability:

$$\mathcal{E}F(t) = \mathcal{E}F(r(t))P(t|r(t)) \tag{4}$$

Using these equations, we can measure for each observed subtree in the corpus, the difference between observed frequency and expected frequency. This will give high values for overrepresented and frequent constructions in the corpus, such as subtrees corresponding to *revenues rose CD % to \$ CD million from \$ CD million last year*, *details weren't disclosed*, *NP-SUBJ declined to comment* and contracted and negated auxiliaries such as *won't*, *can't* and *don't*. The top-10 overrepresented subtrees in the WSJ20-corpus are given in figure 1.

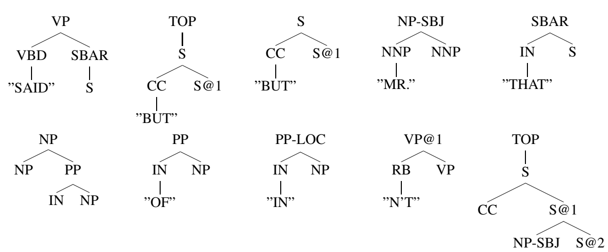

Figure 1: Top-10 overrepresented subtrees (excluding subtrees with punctuation) in sentences of length ≤ 20, including punctuation, in sections 2-21 of the WSJ-corpus (transformed to Chomsky Normal Form, whereby newly created nonterminals are marked with an @). Measured are the deviations from the expected frequencies according to the treebank PCFG (of this selection), as in equation (4) but with $\mathcal{E}F(r(t))$ replaced by the empirical frequency $o(r(t))$. Observed frequencies are (deviations between brackets): 461 (+408.2), 554 (+363.8), 556 (+361.7), 479 (+348.2), 332 (+314.3), 415 (+313.3), 460 (+305.1), 389 (+283.0), 426 (+277.2), 295 (+266.1).

Of course, there are also many subtrees that occur much less frequently than the grammar predicts, such as for instance subtrees corresponding to infrequent or non-occurring variations of the frequent ones, e.g. *revenues rose CD from \$ CD million from \$ CD million*. Underrepresented subtrees found in the WSJ20 corpus, include (VP (VBZ "IS") NP)), which occurs only once, even though it is predicted 152.7 times more often (in all other VP's with "IS", the NP is labeled NP-PRD); and (PP (IN "IN") NP)), which occurs 38 times but is expected 121.0 times more often (IN NP-constructions are usually labeled PP-LOC).

Given such statistics, how do we improve the grammar such that it better models the data? PCFG enrichment models (Klein and Manning, 2003; Schmid, 2006) split (and merge) nonterminals; in automatic enrichment methods (Prescher, 2005; Petrov et al., 2006) these transformations are performed so as to maximize data likelihood (under some constraints). The treebank PCFG-distribution thereby changes, such that the deviations from figure 1 mostly disappear. For instance, the overrepresentation of "but" as the sentence-initial CC in the second and third subtree of that figure, is dealt with in (Schmid, 2006) by splitting the CC-category into CC/BUT and CC/AND. However, also when a range of such transformations is applied, some subtrees are still greatly overrepresented. Figure 2 gives the top-10 overrepresented subtrees of the same treebank, enriched with Schmid's enrichment program `tmod`.

In DOP, larger subtrees can be explicitly represented as units. This is the approach we take in this paper, which involves switching from PCFGs to PTSGs. However, we cannot simply add overrepresented trees to the treebank PCFG; as is clear from figure 2, many of the overrepresented subtrees are in fact spurious variations of the same constructions (e.g. "\$ CD million", "a JJ NN"). To reach our goal of finding the minimal set of subtrees that accurately models the empirical distribution over trees, we will thus need to consider a series of PTSGs, find the subtrees that are still overrepresented and adapt the grammar accordingly.

4 Deviations from an PTSG distribution

4.1 Expected Frequencies: An Example

Once we allow T to contain elementary trees of depth larger than 1, the equations above become more difficult. The reason is that now multiple derivations may give rise to the same parse tree, and,

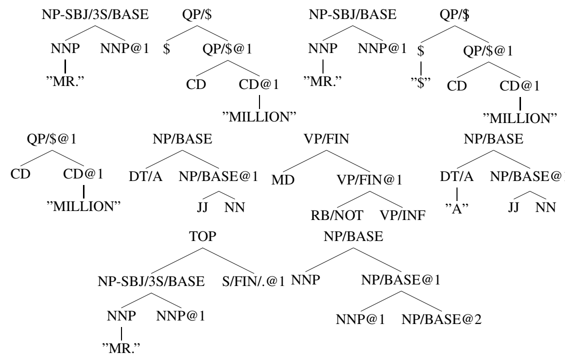

Figure 2: Top-10 overrepresented subtrees (excluding subtrees with punctuation) in the WSJ20 corpus, enriched with the `tmod` program (Schmid, 2006). Empirical frequencies are as follows (deviations between brackets): 262 (+207.6), 235 (+158.4) 207 (+156.4), 228 (+153.5), 237 (+141.0), 190 (+134.2), 153 (+126.5), 166 (+117.8), 139 (+110.0), 111 (+103.8).

as a corrolary, a specific subtree can emerge in many different ways. Consider an PTSG that consists of all subtrees of the trees t_1, t_2 and t_3 in figure 3, and the expected frequency of the subtree t^*.

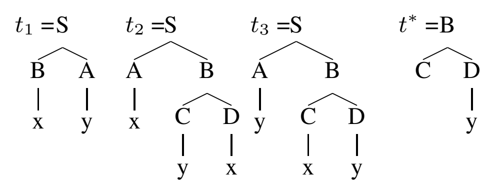

Figure 3: Three example treebank trees and the focal subtree

It is clear that t^* might arise in many different ways. For instance, it emerges in the derivation with elementary trees $\tau_1 \circ \tau_4 \circ \tau_5$ from figure 4, but also in derivations $\tau_2 \circ \tau_4$ and $\tau_3 \circ \tau_5$. Note that in none of these derivations elementary tree t^* itself was used.

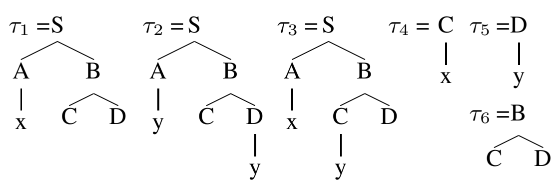

Figure 4: Some elementary trees extracted from the trees in fig 3

4.2 Expected Frequency: Usage & Occurrence

Hence, when using PTSGs, we need to distinguish between the *expected usage frequency* of an elementary tree (written as $\mathcal{E}u(\tau)$), and the *expected occurrence frequency* ($\mathcal{E}o(t)$) of the corresponding subtree. Moreover, not all nonterminal nodes in a derived tree are necessarily "visited" substitution sites. The expected frequency of visiting a nonterminal state X as substitution site depends on the usage frequencies:

$$\mathcal{E}F(X) = \begin{cases} N & \text{if } X = S \\ \sum_\tau \mathcal{E}u(\tau)C(X, l(\tau)) & \text{otherwise} \end{cases} \tag{5}$$

Relating usage frequencies to weights is still simple (compare equation 3):

$$\mathcal{E}u(\tau) = \mathcal{E}F(r(\tau))w(\tau) \tag{6}$$

And hence: $w(\tau) = \mathcal{E}u(\tau)/\sum_{\tau':r(\tau)=r(\tau')} \mathcal{E}u(\tau')$.

The expected frequency of a complete tree is not simply a product anymore, but the sum of the different derivation probabilities (where der(t) gives the set of derivations of t):

$$\mathcal{E}o(t) = \sum_{d \in \text{der}(t)} \prod_{\tau \in d} w(\tau) \qquad \text{if } t \in \Theta \tag{7}$$

4.3 Expected Frequency of Arbitrary Subtrees

Most complex is the expected occurrence frequency of an arbitrary subtree t. From the example above it is clear that it is not necessary that the root of t is a substitution site. Analogous to equation (4), we need the expected frequency of arriving at some state σ in the derivation process that is still consistent with expanding to something that contains t, and multiply it with the probability that this expansion indeed happens:

$$\mathcal{E}o(t) = \sum_\sigma \mathcal{E}F(\sigma)P(t|\sigma) \tag{8}$$

To be able to define the states σ, we redefine the set of derivations $der(t)$ of a subtree t, such that the derivations $der(t^*)$ of our example tree from figure 3 are the following: $d_1 = B \circ \tau_6 \circ \tau_5$, $d_2 = \tau_6 \circ \tau_5$, $d_3 = B \circ t^*$ and $d_4 = t^*$. Only if a derivation starts with a single nonterminal is the root node considered a substitution site. The states σ correspond to the first elements of each of these derivations, i.e. $\langle B, \tau_6, B, t^* \rangle$.

As was clear from the example in section 4.1, we need to consider all supertrees of the trees in the derivation of t for calculating the expected frequency of a state and the probability of expanding from that state to form t. It is useful to distinguish, as do Bod & Kaplan (Bod, 1998, ch. 10) two types of supertree-subtree relations, depending on whether nodes must be removed from the root downward, or from the leaves ("frontier") upward. "Root-subtrees" of t are those subtrees headed by any of t's internal nodes and everything below. "Frontier-subtrees" are those subtrees headed by t's root-node, pruned at any number (≥ 0) of internal nodes. Using $\circ$ to indicate left-most substitution, we can write:

- t_1 is a *root-subtree* of t_1, and t_1 is a *root-subtree* of t_2, if $\exists t_3$, such that $t_3 \circ t_1 = t_2$;
- t_1 is a *frontier-subtree* of t_1, and t_1 is a *frontier-subtree* of t_2, if $\exists t_3 \ldots t_n$, such that $t_1 \circ t_3 \ldots \circ t_n = t_2$.
- t' is the x-frontier-subtree of t, $t' = fs_x(t)$, if x is a set of nodes in t, such that if t is pruned at each $i \in x$ it equals t'.

We use the notation $st(t)$ for the set of subtrees of t, $rs(t)$ for the set of root-subtrees of t and $fs(t)$ for the set of frontier-subtrees of t. Thus defined, the set of all subtrees of t is the set of all frontier-subtrees of all root-subtrees of t: $st(t) = \{t' | \exists t''(t'' \in rs(t) \wedge t' \in fs(t''))$. We further define the sets of root-supertrees, frontier-supertrees, and supertrees as follows: (i) $\widehat{fs_x}(t) = \{t' | t = fs_x(t')\}$, (ii) $\widehat{fs}(t) = \{t' | t \in fs(t')\}$ (iii) $\widehat{st}(t) = \{t' | t \in st(t')\}$.

If there are only terminals in the yield of t, the expected frequency of a state σ is now simply the sum of the expected usage frequencies of those elementary trees that have σ at their *frontier* (i.e. that σ is a root-subtree of):

$$\mathcal{E}F(\sigma) = \sum_{\tau' : \sigma \in rs(\tau')} \mathcal{E}u(\tau') \qquad \text{if } l(t) \subset V_t \tag{9}$$

If there are nonterminals in the yield of t, as in the example, we need to also consider elementary trees that have these nonterminals already expanded. To see why, consider again the example of section 4.1 and check that also elementary tree τ_3 contributes to the expected frequency of t^*. If we take this into account, and write $nt(t)$ for the nonterminal nodes in the yield of t, the final expression for the expected frequency of state σ becomes:

$$\mathcal{E}F(\sigma) = \sum_{\tau \in \widehat{fs}(\sigma)} \sum_{\tau' \in \widehat{rs_{nt(t)}}(\tau)} \mathcal{E}u(\tau') \tag{10}$$

Finally, the probability of expanding a state σ such that t emerges is again simplest if t has no nonterminals as leaves. Remember that a state σ was the first element of a derivation of t; the probability of expanding to t is simply the product of the weights of the remaining elementary trees in the derivation (if states are unique for a derivation):

$$P(t|\sigma) = \prod_{\tau \in \text{rest}(d)} w(\tau) \qquad \text{if } l(t) \subset V_t \quad (11)$$

If there are nonterminals among the leaves of t, however, we need again to sum over possible expansions at those nonterminal leaves:

$$P(t|\sigma) = \prod_{\tau \in \text{rest}(d)} \sum_{\tau' \in \widehat{fs_{x(t)}}(\tau)} w(\tau') \qquad (12)$$

Substituting equations (9) and (12) into equation (8) gives a general expression for the expected occurrence frequency of an arbitary subtree t:

$$\mathcal{E}o(t) = \sum_{d \in der(t)} \Bigg(\prod_{\substack{\tau \in \\ \text{rest}(d)}} \sum_{\tau' \in \widehat{fs_{x(t)}}(\tau)} w(\tau') \sum_{\substack{\tau \in \\ \widehat{rs}(\text{first}(d))}} \sum_{\tau' \in \widehat{fs_{x(t)}}(\tau)} \mathcal{E}u(\tau') \Bigg). \qquad (13)$$

5 Minimizing deviations: estimation

The equations just derived can be used to learn an PTSG from a treebank, using an estimation procedure we call "push-n-pull" (`pnp`). This procedure was described in some detail elsewhere (Zuidema, 2006b); here I only sketch the basic idea. Given an initial setting of the parameters (all depth 1 elementary trees at their empirical frequency), the method calculates the expected frequency of all complete and incomplete trees. If a tree t's expected frequency $\mathcal{E}o(t)$ is higher than its observed frequency $o(t)$, the method subtracts the difference from the tree's score, and distributes ("pushes") it over the elementary trees involved in all its derivations ($der(t)$). If it is lower, it "pulls" the difference from all its derivations.

The "score" of an elementary tree τ is the algorithm's estimate of the usage frequency $u(\tau)$. The amounts of score that are pushed or pulled are capped by the requirement that $\forall \tau\ u(\tau) \geq 0$; moreover, the learning rate parameter γ determines the fraction of the expected-observed difference that is actually pushed or pulled. Finally, the method includes a bias (B) for moving probability mass to smaller elementary trees, to avoid overfitting (its effects become smaller as more data gets observed).

Because smaller elementary trees will be involved in other derivations as well, the push and pull operations will shift probabilities between different parse trees. Suppose a given complete tree is the only tree with nonzero frequency of all trees that can be built from the same components. This tree will continue to "pull" until it has in fact reached its appropriate frequency. Similarly, if a given tree does have zero observed frequency, it will continue to leak score to other derivations with the same components.

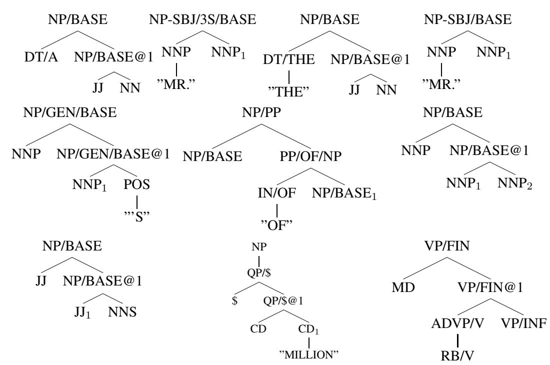

Figure 5: Top-10 elementary trees of depth>1, excluding those with punctuation, from running `pnp` on the enriched WSJ20.

The output of the push-n-pull algorithm is an PTSG, with the same set of elementary trees as the DOP models of Bod (1998; 2001). This set is very large. However, unlike those existing DOP models, the score distribution over these elementary trees is extremely skewed: relatively few trees receive high scores, and there is a very long tail of trees with low scores. In Zuidema (2006b) we give a qualitative analysis of the subtrees with the highest scores as induced from the ATIS treebank, which include many of its frequent constructions including *show me NP*, *I would like NP*, *flights from NP to NP*. The top-10 larger elementary trees that result from running `pnp` on a randomly selected trainset of about 8000 sentences of the Dutch OVIS treebank (Veldhuijzen van Zanten et al., 1999), can be glossed as: *Yes, from NP to NP, No thank you very much, I want to VP-INF,* *No thank you, I want PP to VP-INF, I want PP, I want Prep NP-LOC Prep NP-LOC, Yes please, At CD o'clock*. In figure 5 we give the top-10 elementary trees resulting from the WSJ20-corpus.

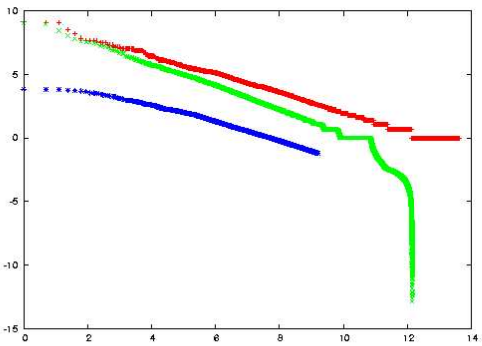

Figure 6: Log-log curves of (i) subtree frequencies against rank (for 10^6 subtrees from WSJ20), (ii) pnp-scores against rank, and (iii) the same for the top-10000 depth>1-subtrees.

Figure 6 shows some characteristics of this last grammar. Shown are log-log plots, such as commonly used to visualise the Zipf-distributions in natural language. The top curve plots log(frequency) against log(rank) for each subtree of the trees in the corpus, which shows the approximate Zipfian behavior. The second curve from above plots the log(score) against log(rank) for these same subtrees. As can be observed, the score-distribution follows the frequency distribution only for the most frequent subtrees (all of depth 1), but then deviates from it downwards. The bottom curve – an almost straight line in this log-log space – gives the log(score) vs log(rank) of subtrees with a depth>1.

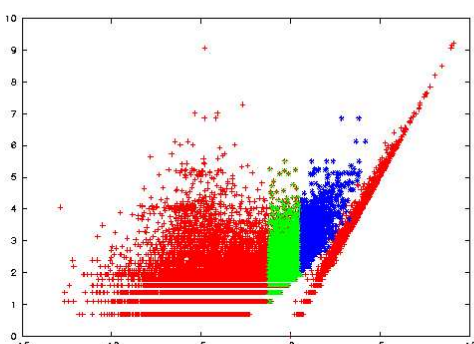

Figure 7: Subtree frequencies against pnp-scores, including subsets pnp1000 (dark/blue) and pnp10000 (light/green).

Figure 7 further illustrates the difference between the score- and the frequency-distributions, by plotting for each subtree, log-frequency (y-axis) against log-score (x-axis). The subtrees clearly fall into two categories: those where the scores correlate strongly with frequency (the depth 1 subtrees) and the larger subtrees that vary greatly in how strong scores correlate with frequency. Only larger subtrees that receive relatively high scores should be used in parsing.

Weights are proportional to subtree-frequencies in the DOP1 and related "maximalist" models. The differences between the frequency and score-distributions thus illustrate a very important difference between maximalist and parsimonious DOP. The characteristics of the score distribution allow P-DOP to throw away most of the subtrees without significantly affecting the distribution over complete parse trees that the grammar defines. This is the approach we take for evaluating parsing performance: we take as our baseline the treebank PCFG, and then add the n larger elementary trees with the highest scores from our induced PTSG.

6 Parsing Results

For our parsing results we use BitPar, a fast and freely available general PCFG parser (Schmid, 2004). In our first experiments we used the OVIS corpus, with semantic tags and punctuation removed, and all trees (train- and testset) binarized. As a baseline experiment, we read off the treebank PCFG as decribed in section 2. The recall, precision and complete match results are in table 1, labeled tb-pcfg. For comparison, we also show the results obtained with two versions of the DOP model, DOP1 (Bod, 1998) and DOP* (Zollmann and Sima'an, 2005) on the same treebank.

We ran the pnp program as described above on the trainset, with parameters $B = 1.0$, $\gamma = 0.1$ and $d = 4$. This run yielded a single PTSG that was used in 4 parsing runs. For these experiments, we added increasingly many of the depth>1 elementary trees from the PTSG, with minimum scores of 7.0, 1.0, 0.5, and 0.075. The added elementary trees were first converted to PCFG rules, by labeling all internal nodes with a unique address label and reading off the CFG-productions. Each rule received a score equal to the score of the elementary tree it derived from. A copy of each rule, with the label removed, was also added with a negative score, BitPar automatically sums (and substracts) and normalizes the frequency information provided with each rule. BitPar was then run on the testset sentences, with the option to output the n best parses with $n = 10$ by default. These parses were then read in in a postprocessing program, which removes address labels, sums probabilities of equivalent parses and outputs the most probable parse for each sentence (this is the same approximation of MPP, albeit with smaller n, as used in most of Bod's DOP results). The results of these experiments are also in table 1, labeled pnp$N$, where N is the number of elementary trees added.

model	# rules	LR	LP	CM
tb-pcfg	3000	93.45	95.5	85.84
DOP1	1.4×10^6			(87.55)
DOP*	($<$ 50000)			(87.7)
pnp100	3000+100	93.63	95.65	86.55
pnp763	3000+763	93.5	95.52	86.75
pnp1517	3000+1517	93.78	95.83	87.36
pnp11411	3000+11411	**94.26**	**96.4**	**87.77**

Table 1: Results on the Dutch OVIS tree bank, with semantic tags and punctuation removed. Reported are evalb scores on a random testset of 1000 sentences (a second testset of 1000 sentences is kept for later evaluations). The trainset for both the treebank grammar and the pnp program consists of the remaining 8049 trees. Coverage in all cases in 989 sentences out of 1000. Results in brackets are from Zollman & Sima'an, 2005, using a different train-test set split.

As these experiments show, adding larger elementary trees from the induced PTSG, in order of their assigned scores, monotonously increases the parse accuracy of the treebank PCFG. Although the final grammar is at least 5 times larger than the original treebank PCFG, and the parser therefore slower, the grammar is orders of magnitude smaller than the corresponding maximalist DOP models and shows comparable parse accuracy.

For a second set of parsing experiments, we used the WSJ portion of the Penn Tree Bank (Marcus et al., 1993) and Helmut Schmid's enrichment program tmod (Schmid, 2006). Schmid's program enriches nonterminal labels in the treebank, using features inspired by (Klein and Manning, 2003). After enrichment, Schmid obtained excellent parsing scores with the treebank PCFG. In table 2, as model tb-pcfg, we give our baseline results. These are slightly lower than Schmid's, for two reasons: (i) our implementation ignores the upper/lower case distinction, and (ii) we do not use Schmid's word class automaton for unknown words (the only smoothing used is the built-in feature of the BitPar parser, which extracts an open-class-tag file from the lexicon file). Because our interest here is in the principles of enrichment we have not attempted to adapt these techniques for our implementation.

As before, we ran the pnp program on the trainset, the enriched sections 2-21 of the WSJ. For computational reasons, pnp is only run on trees with a yield of length (including punctuation) $\leq$ 20. This run, which took several days on a machine with 1.5Gb RAM, again produced a very large PTSG, from which we extracted the 1000 and 10000 depth>1 elementary trees with the highest scores for parsing experiments. Parsing and postprocessing is performed as before, with the MPP approximated from the best $n = 20$ parses. Results from these experiments are shown in table 2, as models pnp1000 and pnp10000. With a small number of added trees, we see a small drop in the parsing performance, which we interpret as evidence that our additions somewhat disturb the nicely tuned probability distribution of the treebank PCFG without providing many advantages, because the most frequent constructions have already been addressed in the manual PCFG enrichment. However, with 10000 added subtrees we see an increase in parse accuracy, providing evidence that pnp has learned potential enrichments that go beyond the manual enrichment.

model	LR	LP	F_1	CM
tb-pcfg	83.27	83.53	83.40	26.58
pnp1000	83.20	83.47	83.33	26.70
pnp10000	**83.56**	**83.99**	**83.77**	**26.93**

Table 2: Results on the WSJ section of the Penn Tree Bank, where nonterminals are enriched with features using Helmut Schmid's tmod program (Schmid, 2006). Reported are evalb scores (ignoring punctuation) on 1699 sentences ≤ 100, including punctuation, from section 22. Sections 02-21 were the train set for the treebank PCFG; only trees with a yield (including punctuation) of length ≤ 20 were used for the pnp program. Coverage in all cases is 1691 (excluding failed parses gives $F_1 = 85.19$ for the tb-pcfg-baseline, and 85.54 for pnp10000).

In figure 8(left) we plot the difference in parse accuracy between the treebank PCFG and our best model per testset sentence. To make the plot more informative, the sentences are ordered by increasing difference. Hence, on the left are sentences where the treebank PCFG scores better, and at the right the sentence where pnp10000 scores best. As is clear from this graph, for most sentences there is no difference, but there are small and about equally sized subsets of sentences for which one or the other model scores better. We have briefly analysed these sentences, but not found a clear pattern. In figure 8(right) we plot in a similar way the difference in log-likelihood that the parsers assign to each sentence. Here we see a clear pattern: only very few sentences receive slightly higher likelihood under the PCFG model. For a good portion of the sentences, however, the pnp10000 model assigns them somewhat and in some cases much higher likelihood. The highest likelihood gains are due to a small number of frequent multiword expressions, such as "New York Stock Exchange Composite Trading", which P-DOP treats as a unit; all of the other gains in likelihood are also due to the use of depth>1 elementary trees, including some non-contiguous constructions such as *revenues rose CD % to $ CD million from $ CD million*.

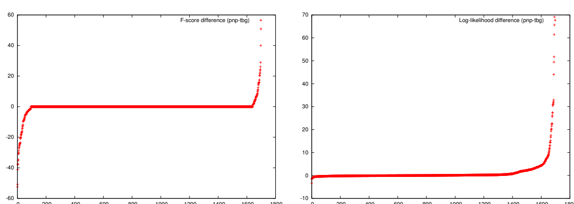

Figure 8: Per sentence difference in f-score (left) and log-likelihood (right) of the sentences of WSJ section 22. The x-axis gives the sentence-rank when sentences are ordered from small to large on y-axis value.

7 Discussion and Conclusions

We set out to develop a parsimonious approach to Data-Oriented Parsing, where all subtrees can potentially become units in a probabilistic grammar but only if the statistics require it. The grammars resulting from our algorithm are orders of magnitude smaller than those used in Bod's maximalist DOP. Although our parsing results are not yet at the level of the best results obtained by Bod, our results indicate that we are getting closer and that we already induce linguistically more plausible grammars.

Could P-DOP eventually not only be more efficient, but also more accurate than maximalist DOP models? Bod has argued that the explanation for DOP's excellent results is that it takes into account all possible dependencies between productions in a tree, and not just those from an a-priori chosen subset (e.g. lexical, head, parent features). Non-head dependencies in non-contiguous natural language constructions, like *more ... than*, as in *more freight than cargo*, are typically excluded in the enrichment/conditioning approaches discussed in section 2. Bod wants to include any dependency a priori, and then "let the statistics decide".

Although the inclusion of all dependencies must somehow explain the performance difference between Bod's best generative model and manually enriched PCFG models, this explanation is not entirely satisfactory. Zuidema (2006a) shows that also the estimator (Bod, 2003) uses is biased and inconsistent, and will, even in the limit of infinite data, not correctly identify many possible distributions over trees. This is not just a theoretical problem. For instance, in the Penn Tree Bank the construction *won't VP* is annotated as *(VP (MD wo) (RB n't) VP)*. There is a strong dependency between the two morphemes: *wo* doesn't exist as an independent word, and strongly predicts *n't*. However, Bod's estimator will continue to reserve probability mass for other combinations with the same POS-tags such as *wo not*, even with an infinite data set only containing *will not* and *wo n't*. Because in parsing the strings are given, this particular example will not harm the parse accuracy results. The example might be diagnostic for other cases that do, however, and certainly will have impact when DOP is used as language model. P-DOP, in contrast, does converge to grammars that treat *won't* as a single unit.

The exact relation of P-DOP to other DOP models, including S-DOP (Bod, 2003), Backoff-DOP (Sima'an and Buratto, 2003), DOP* (Zollmann and Sima'an, 2005) and ML-DOP (Bod, 2006; based on Expectation Maximization) and not dissimilar automatic enrichment models such as (Petrov et al., 2006), remains a topic for future work.

Acknowledgments Funded by the Netherlands Organisation for Scientific Research (EW), project nr. 612.066.405; many thanks to Reut Tsarfaty, Remko Scha, Rens Bod and three anonymous reviewers for their comments.

References

Rens Bod. 1998. *Beyond Grammar: An experience-based theory of language*. CSLI, Stanford, CA.

Rens Bod. 2001. What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings ACL-2001*.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings EACL'03*.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. *Proceedings ACL-COLING'06*.

Eugene Charniak. 1996. Tree-bank grammars. Technical report, Department of Computer Science, Brown University

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the fourteenth national conference on artificial intelligence*, Menlo Park. AAAI Press/MIT Press.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings ACL 2002*.

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings ACL'97*, pages 16–23.

Joshua Goodman. 1996. Efficient algorithms for parsing the DOP model. In *Proceedings EMNLP*, pages 143–152.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Mark Johnson. 2002. The DOP estimation method is biased and inconsistent. *Computational Linguistics*, 28(1):71–76.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings ACL'03*.

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

T. Matzuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings ACL'05*, pages 75–82.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings ACL-COLING'06*, pages 443–440.

Detlef Prescher. 2005. Inducing head-driven PCFGs with latent heads: Refining a tree-bank grammar for parsing. In *Proceedings ECML'05*.

Remko Scha. 1990. Taaltheorie en taaltechnologie; competence en performance. In R. de Kort and G.L.J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, pages 7–22. LVVN, Almere, the Netherlands. English translation at http://iaaa.nl/rs/LeerdamE.html.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings COLING 2004*.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of COLING-ACL 2006*.

Khalil Sima'an. 2002. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151.

Khalil Sima'an and Luciano Buratto. 2003. Backoff parameter estimation for the DOP model. In *Proceedings of the 14th European Conference on Machine Learning (ECML'03, Cavtat-Dubrovnik, Croatia*, number 2837 in Lecture Notes in Artificial Intelligence, pages 373–384. Springer Verlag, Berlin, Germany.

Gert Veldhuijzen van Zanten, Gosse Bouma, Khalil Sima'an, Gertjan van Noord, and Remko Bonnema. 1999. Evaluation of the NLP components of the OVIS2 spoken dialogue system. In van Eynde, Schuurman, and Schelkens, editors, *Computational Linguistics in the Netherlands 1998*, pages 213–229. Rodopi, Amsterdam.

Andreas Zollmann and Khalil Sima'an. 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, 10(2/3):367–388.

Willem Zuidema. 2006a. Theoretical evaluation of estimation methods for Data-Oriented Parsing. In *Proceedings EACL 2006 (Conference Companion)*, pages 183–186. Association for Computational Linguistics. Erratum on http://staff.science.uva.nl/˜jzuidema/research.

Willem Zuidema. 2006b. What are the productive units of natural language grammar? A DOP approach to the automatic identification of constructions. In *Proceedings of the 10th International Conference on Computational Natural Language Learning (CONLL-X)*.