# Unsupervised Parsing with U-DOP

**Rens Bod**
School of Computer Science
University of St Andrews
North Haugh, St Andrews
KY16 9SX Scotland, UK
rb@dcs.st-and.ac.uk

## Abstract

We propose a generalization of the supervised DOP model to unsupervised learning. This new model, which we call U-DOP, initially assigns all possible unlabeled binary trees to a set of sentences and next uses all subtrees from (a large subset of) these binary trees to compute the most probable parse trees. We show how U-DOP can be implemented by a PCFG-reduction technique and report competitive results on English (WSJ), German (NEGRA) and Chinese (CTB) data. To the best of our knowledge, this is the first paper which accurately bootstraps structure for Wall Street Journal sentences up to 40 words obtaining roughly the same accuracy as a binarized *supervised* PCFG. We show that previous approaches to unsupervised parsing have shortcomings in that they either constrain the lexical or the structural context, or both.

## 1 Introduction

How can we learn syntactic structure from unlabeled data in an unsupervised way? The importance of unsupervised parsing is nowadays widely acknowledged. While supervised parsers suffer from shortage of hand-annotated data, unsupervised parsers operate with unlabeled raw data, of which unlimited quantities are available. During the last few years there has been considerable progress in unsupervised parsing. To give a brief overview: van Zaanen (2000) achieved 39.2% unlabeled f-score on ATIS word strings by a sentence-aligning technique called ABL. Clark (2001) reports 42.0% unlabeled f-score on the same data using distributional clustering, and Klein and Manning (2002) obtain 51.2% unlabeled f-score on ATIS part-of-speech strings using a constituent-context model called CCM. Moreover, on Penn Wall Street Journal p-o-s-strings ≤ 10 (WSJ10), Klein and Manning (2002) report 71.1% unlabeled f-score. And the hybrid approach of Klein and Manning (2004), which combines a constituency and a dependency model, leads to a further increase of 77.6% f-score.

Although there has thus been steady progress in unsupervised parsing, all these approaches have shortcomings in that they either constrain the lexical or the structural context that is taken into account, or both. For example, the CCM model by Klein and Manning (2005) is said to describe "all contiguous subsequences of a sentence" (Klein and Manning 2005: 1410). While this is a very rich lexical model, it is still limited in that it neglects dependencies that are *non*-contiguous such as between *more* and *than* in "*BA carried more people than cargo*". Moreover, by using an "all-*substrings*" approach, CCM risks to under-represent *structural* context. Similar shortcomings can be found in other unsupervised models.

In this paper we will try to directly model structural as well as lexical context without constraining any dependencies beforehand. An approach that may seem apt in this respect is an *all-subtrees* approach (e.g Bod 2003; Goodman 2003; Collins and Duffy 2002). Subtrees can model both contiguous and non-contiguous lexical dependencies (see section 2) and they also model constituents in a hierarchical context. Moreover, we can view the all-subtrees approach as a generalization of Klein and Manning's all-substrings approach and van Zaanen's ABL model.

In the current paper, we will use the all-subtrees approach as proposed in Data-Oriented

*Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)*,
pages 85–92, New York City, June 2006.

Parsing or DOP (Bod 1998). We will generalize the supervised version of DOP to unsupervised parsing. The key idea of our approach is to initially assign all possible unlabeled binary trees to a set of given sentences, and to next use counts of all subtrees from (a large random subset of) these binary trees to compute the most probable parse trees. To the best of our knowledge, such a model has never been tried out. We will refer to this unsupervised DOP model as *U-DOP*, while the supervised DOP model (which uses hand-annotated trees) will be referred to as *S-DOP*. Moreover, we will continue to refer to the general approach simply as *DOP*.

U-DOP is not just an engineering approach to unsupervised learning but can also be motivated from a cognitive perspective (Bod 2006): if we don't have a clue which trees should be assigned to sentences in the initial stages of language acquisition, we can just as well assume that initially all trees are possible. Only those (sub)trees that partake in computing the most probable parse trees for new sentences are actually "learned". We have argued in Bod (2006) that such an integration of unsupervised and supervised methods results in an integrated model for language learning and language use.

In the following we will first explain how U-DOP works, and how it can be approximated by a PCFG-reduction technique. Next, in section 3 we discuss a number of experiments with U-DOP and compare it to previous models on English (WSJ), German (NEGRA) and Chinese (CTB) data. To the best of our knowledge, this is the first paper which bootstraps structure for WSJ sentences up to 40 words obtaining roughly the same accuracy as a binarized *supervised* PCFG. This is remarkable since unsupervised models are clearly at a disavantage compared to supervised models which can literally reuse manually annotated data.

## 2 Unsupervised data-oriented parsing

At a general level, U-DOP consists of the following three steps:

1. Assign all possible binary trees to a set of sentences
2. Convert the binary trees into a PCFG-reduction of DOP
3. Compute the most probable parse tree for each sentence

Note that in unsupervised parsing we do not need to split the data into a training and a test set. In this paper, we will present results both on entire corpora and on 90-10 splits of such corpora so as to make our results comparable to a *supervised* PCFG using the treebank grammars of the same data ("*S-PCFG*").

In the following we will first describe each of the three steps given above where we initially focus on inducing trees for p-o-s strings for the WSJ10 (we will deal with other corpora and the much larger WSJ40 in section 3). As shown by Klein and Manning (2002, 2004), the extension to inducing trees for words instead of p-o-s tags is rather straightforward since there exist several unsupervised part-of-speech taggers with high accuracy, which can be combined with unsupervised parsing (see e.g. Schütze 1996; Clark 2000).

### Step 1: Assign all binary trees to p-o-s strings from the WSJ10

The WSJ10 contains 7422 sentences ≤ 10 words after removing empty elements and punctuation. We assigned all possible binary trees to the corresponding part-of-speech sequences of these sentences, where each root node is labeled *S* and each internal node is labeled *X*. As an example, consider the p-o-s string NNS VBD JJ NNS, which may correspond for instance to the sentence *Investors suffered heavy losses*. This string has a total of five binary trees shown in figure 1 -- where for readability we add words as well.

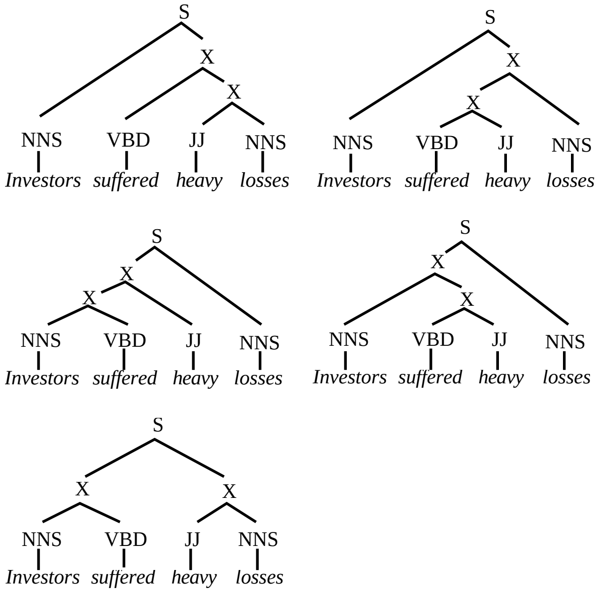

Figure 1. All binary trees for NNS VBD JJ NNS (*Investors suffered heavy losses*)

The total number of binary trees for a sentence of length $n$ is given by the Catalan number $C_{n-1}$, where $C_n = (2n)!/((n+1)!\,n!)$. Thus while a sentence of 4 words has 5 binary trees, a sentence of 8 words has already 429 binary trees, and a sentence of 10 words has 4862 binary trees. Of course, we can represent the set of binary trees of a string in polynomial time and space by means of a chart, resulting in a chart-like parse forest if we also include pointers. But if we want to extract rules or subtrees from these binary trees -- as in DOP -- we need to unpack the parse forest. And since the total number of binary trees that can be assigned to the WSJ10 is almost 12 million, it is doubtful whether we can apply the unrestricted U-DOP model to such a corpus.

However, for longer sentences the binary trees are highly redundant. In these larger trees, there are many rules like $X \rightarrow XX$ which bear little information. To make parsing with U-DOP possible we therefore applied a simple heuristic which takes random samples from the binary trees for sentences ≥ 7 words before they are fed to the DOP parser. These samples were taken from the distribution of all binary trees by randomly choosing nodes and their expansions from the chart-like parse forests of the sentences (which effectively favors trees with more frequent subtrees). For sentences of 7 words we randomly sample 60% of the trees, and for sentences of 8, 9 and 10 words we sample respectively 30%, 15% and 7.5% of the trees. In this way, the set of remaining binary trees contains $8.23 * 10^5$ trees, which we will refer to as the *binary tree-set*. Although it can happen that the correct tree is deleted for some sentence in the binary tree-set, there is enough redundancy in the tree-set such that either the correct binary tree can be generated by other subtrees or that a remaining tree only minimally differs from the correct tree. Of course, we may expect better results if *all* binary trees are kept, but this involves enormous computational resources which will be postponed to future research.

### Step 2: Convert the trees into a PCFG-reduction of DOP

The underlying idea of U-DOP is to take all subtrees from the binary tree-set to compute the most probable tree for each sentence. Subtrees from the trees in figure 1 include for example the subtrees in figure 2 (where we again added words for readability). Note that U-DOP takes into account both contiguous and non-contiguous substrings.

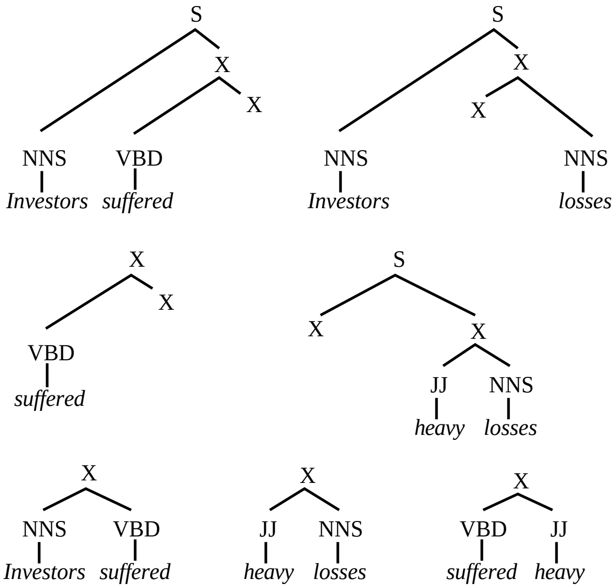

Figure 2. Some subtrees from the binary trees for NNS VBD JJ NNS given in figure 1

As in the supervised DOP approach (Bod 1998), U-DOP parses a sentence by combining corpus-subtrees from the binary tree-set by means of a leftmost node substitution operation, indicated as ∘. The probability of a parse tree is computed by summing up the probabilities of all derivations producing it, while the probability of a derivation is computed by multiplying the (smoothed) relative frequencies of its subtrees. That is, the probability of a subtree $t$ is taken as the number of occurrences of $t$ in the binary tree-set, $|\,t\,|$, divided by the total number of occurrences of all subtrees $t'$ with the same root label as $t$. Let $r(t)$ return the root label of $t$:

$$P(t) \;=\; \frac{|\,t\,|}{\sum_{t':\, r(t')=r(t)} \;|\,t'\,|}$$

The subtree probabilities are smoothed by applying simple Good-Turing to the subtree distribution (see Bod 1998: 85-87). The probability of a derivation $t_1 \circ ... \circ t_n$ is computed by the product of the probabilities of its subtrees $t_i$:

$$P(t_1 \circ ... \circ t_n) \;=\; \Pi_i \, P(t_i)$$

Since there may be distinct derivations that generate the same parse tree, the probability of a parse tree $T$

is the sum of the probabilities of its distinct derivations. Let $t_{id}$ be the $i$-th subtree in the derivation $d$ that produces tree $T$, then the probability of $T$ is given by

$$P(T) \;=\; \Sigma_d \Pi_i \, P(t_{id})$$

As we will explain under step 3, the most probable parse tree of a sentence is estimated by Viterbi $n$-best summing up the probabilities of derivations that generate the same tree.

It may be evident that had we only the sentence *Investors suffered heavy losses* in our corpus, there would be no difference in probability between the five parse trees in figure 1, and U-DOP would not be able to distinguish between the different trees. However, if we have a different sentence where JJ NNS (*heavy losses*) appears in a different context, e.g. in *Heavy losses were reported*, its covering subtree gets a relatively higher frequency and the parse tree where *heavy losses* occurs as a constituent gets a higher total probability than alternative parse trees. Of course, it is left to the experimental evaluation whether *non*-constituents ("distituents") such as VBD JJ will be ruled out by U-DOP (section 3).

An important feature of (U-)DOP is that it considers counts of subtrees of a wide range of sizes: everything from counts of single-level rules to entire trees. A disadvantage of the approach is that an extremely large number of subtrees (and derivations) must be taken into account. Fortunately, there exists a rather compact PCFG-reduction of DOP which can also be used for U-DOP (Goodman 2003). Here we will only give a short summary of this PCFG-reduction. (Collins and Duffy 2002 show how a tree kernel can be used for an all-subtrees representation, which we will not discuss here.)

Goodman's reduction method first assigns every node in every tree a unique number which is called its address. The notation $A@k$ denotes the node at address $k$ where $A$ is the nonterminal labeling that node. A new nonterminal is created for each node in the training data. This nonterminal is called $A_k$. 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 nonterminal $A$, that is $a = \Sigma_j a_j$. Goodman then gives a small PCFG with the following property: for every subtree in the training corpus headed by $A$, the grammar will generate an isomorphic subderivation with probability $1/a$. For a node $A@j(B@k, C@l)$, the following eight PCFG rules in figure 3 are generated, where the number in parentheses following a rule is its probability.

$$\begin{array}{llll}
A_j \rightarrow BC & (1/a_j) & A \rightarrow BC & (1/a) \\
A_j \rightarrow B_kC & (b_k/a_j) & A \rightarrow B_kC & (b_k/a) \\
A_j \rightarrow BC_l & (c_l/a_j) & A \rightarrow BC_l & (c_l/a) \\
A_j \rightarrow B_kC_l & (b_kc_l/a_j) & A \rightarrow B_kC_l & (b_kc_l/a)
\end{array}$$

Figure 3. PCFG-reduction of DOP

In this PCFG reduction, $b_k$ represents the number of subtrees headed by the node $B@k$, and $c_l$ refers to the number of subtrees headed by the node $C@l$. Goodman shows by simple induction that his construction produces PCFG derivations isomorphic to (U-)DOP derivations with equal probability (Goodman 2003: 130-133). This means that summing up over derivations of a tree in DOP yields the same probability as summing over all the isomorphic derivations in the PCFG.[1]

The PCFG-reduction for U-DOP is slightly simpler than in figure 3 since the only labels are $S$ and $X$, and the part-of-speech tags. For the tree-set of $8.23 * 10^5$ binary trees generated under step 1, Goodman's reduction method results in a total number of $14.8 * 10^6$ distinct PCFG rules. While it is still feasible to parse with a rule-set of this size, it is evident that our approach can deal with longer sentences only if we further reduce the size of our binary tree-set.

It should be kept in mind that while the probabilities of all parse trees generated by DOP sum up to 1, these probabilities do not converge to the "true" probabilities if the corpus grows to infinity (Johnson 2002). In fact, in Bod et al. (2003) we showed that the most probable parse tree as defined above has a tendency to be constructed by the *shortest derivation* (consisting of the fewest and thus largest subtrees). A large subtree is overruled only if the combined relative frequencies of smaller subtrees yields a larger score. We refer to Zollmann and Sima'an (2005) for a recently proposed estimator that *is* statistically consistent (though it is not yet known how this estimator performs on the WSJ) and to Zuidema (2006) for a theoretical comparison of existing estimators for DOP.

[1] As in Bod (2003) and Goodman (2003: 136), we additionally use a correction factor to redress DOP's bias discussed in Johnson (2002).

### Step 3: Compute the most probable parse tree for each WSJ10 string

While Goodman's reduction method allows for efficiently computing the most probable derivation for each sentence (i.e. the Viterbi parse), it does not allow for an efficient computation of (U-)DOP's most probable parse tree since there may be exponentially many derivations for each tree whose probabilities have to be summed up. In fact, the problem of computing the most probable tree in DOP is known to be NP hard (Sima'an 1996). Yet, the PCFG reduction in figure 4 can be used to *estimate* DOP's most probable parse tree by a Viterbi *n*-best search in combination with a CKY parser which computes the *n* most likely derivations and next sums up the probabilities of the derivations producing the same tree. (We can considerably improve efficiency by using *k*-best hypergraph parsing as recently proposed by Huang and Chiang 2005, but this will be left to future research).

In this paper, we estimate the most probable parse tree from the 100 most probable derivations (at least for the relatively small WSJ10). Although such a heuristic does not guarantee that the most probable parse is actually found, it is shown in Bod (2000) to perform at least as well as the estimation of the most probable parse with Monte Carlo techniques. However, in computing the 100 most probable derivations by means of Viterbi it is prohibitive to keep track of all subderivations at each edge in the chart. We therefore use a pruning technique which deletes any item with a probability less than $10^{-5}$ times of that of the best item from the chart.

To make our parse results comparable to those of Klein and Manning (2002, 2004, 2005), we will use exactly the same evaluation metrics for unlabeled precision (UP) and unlabeled recall (UR), defined in Klein (2005: 21-22). Klein's definitions slightly differ from the standard PARSEVAL metrics: multiplicity of brackets is ignored, brackets of span one are ignored and the bracket labels are ignored. The two metrics of UP and UR are combined by the unlabled f-score F1 which is defined as the harmonic mean of UP and UR: F1 = 2*UP*UR/(UP+UR). It should be kept in mind that these evaluation metrics were clearly inspired by the evaluation of *supervised* parsing which aims at mimicking *given* tree annotations as closely as possible. Unsupervised parsing is different in this respect and it is questionable whether an evaluation on a pre-annotated corpus such as the WSJ is the most appropriate one. For a subtle discussion on this issue, see Clark (2001) or Klein (2005).

## 3 Experiments

### 3.1 Comparing U-DOP to previous work

Using the method described above, our parsing experiment with all p-o-s strings from the WSJ10 results in an f-score of 78.5%. We next tested U-DOP on two additional domains from Chinese and German which were also used in Klein and Manning (2002, 2004): the Chinese treebank (Xue et al. 2002) and the NEGRA corpus (Skut et al. 1997). The CTB10 is the subset of p-o-s strings from the Penn Chinese treebank containing 10 words or less after removal of punctuation (2437 strings). The NEGRA10 is the subset of p-o-s strings of the same length from the NEGRA corpus using the supplied converson into Penn treebank format (2175 strings). Table 1 shows the results of U-DOP in terms of UP, UR and F1 compared to the results of the CCM model by Klein and Manning (2002), the DMV dependency learning model by Klein and Manning (2004) together with their combined model DMV+CCM.

| Model | English (WSJ10) | | | German (NEGRA10) | | | Chinese (CTB10) | | |
|---|---|---|---|---|---|---|---|---|---|
| | UP | UR | F1 | UP | UR | F1 | UP | UR | F1 |
| CCM | 64.2 | 81.6 | 71.9 | 48.1 | 85.5 | 61.6 | 34.6 | 64.3 | 45.0 |
| DMV | 46.6 | 59.2 | 52.1 | 38.4 | 69.5 | 49.5 | 35.9 | **66.7** | **46.7** |
| DMV+CCM | 69.3 | 88.0 | 77.6 | 49.6 | 89.7 | 63.9 | 33.3 | 62.0 | 43.3 |
| U-DOP | **70.8** | **88.2** | **78.5** | **51.2** | **90.5** | **65.4** | **36.3** | 64.9 | 46.6 |

Table 1. Results of U-DOP compared to previous models on the same data

Table 1 indicates that our model scores slightly better than Klein and Manning's combined DMV+CCM model, although the differences are small (note that for Chinese the single DMV model scores better than the combined model and slightly better than U-DOP). But where Klein and Manning's combined model is based on both a constituency and a dependency model, U-DOP is, like CCM, only based on a notion of constituency. Compared to CCM alone, the all-subtrees approach employed by U-DOP shows a clear improvement (except perhaps for Chinese). It thus seems to pay off to use all subtrees rather than just all (contiguous) substrings in bootstrapping constituency. It would be interesting to investigate an extension of U-DOP towards dependency parsing, which we will leave for future research. It is also noteworthy that U-DOP does not employ a separate class for non-constituents, so-called distituents, while CCM does. Thus good results can be obtained without keeping track of distituents but by simply assigning all binary trees to the strings and letting the DOP model decide which substrings are most likely to form constituents.

To give an idea of the constituents learned by U-DOP for the WSJ10, table 2 shows the 10 most frequently constituents in the trees induced by U-DOP together with the 10 actually most frequently occurring constituents in the WSJ10 and the 10 most frequently occurring part-of-speech sequences (bigrams) in the WSJ10.

| Rank | Most frequent U-DOP constituents | Most Frequent WSJ10 constituents | Most frequent WSJ10 substrings |
|---|---|---|---|
| 1 | DT NN | DT NN | NNP NNP |
| 2 | NNP NNP | NNP NNP | DT NN |
| 3 | DT JJ NN | CD CD | JJ NN |
| 4 | IN DT NN | JJ NNS | IN DT |
| 5 | CD CD | DT JJ NN | NN IN |
| 6 | DT NNS | DT NNS | DT JJ |
| 7 | JJ NNS | JJ NN | JJ NNS |
| 8 | JJ NN | CD NN | NN NN |
| 9 | VBN IN | IN NN | CD CD |
| 10 | VBD NNS | IN DT NN | NN VBZ |

Table 2. Most frequently learned constituents by U-DOP together with most frequently occurring constituents and p-o-s sequences (for WSJ10)

Note that there are no distituents among U-DOP's 10 most frequently learned constituents, whilst the third column shows that distituents such as IN DT or DT JJ occur very frequently as substrings in the WSJ10. This may be explained by the fact that (the constituent) DT NN occurs more frequently as a substring in the WSJ10 than (the distituent) IN DT, and therefore U-DOP's probability model will favor a covering subtree for IN DT NN which consists of a division into IN *X* and DT NN rather than into IN DT and *X* NN, other things being equal. The same kind reasoning can be made for a subtree for DT JJ NN where the constituent JJ NN occurs more frequently as a substring than the distituent DT JJ. Of course the situation is somewhat more complex in DOP's sum-of-products model, but our argument may illustrate why distituents like IN DT or DT JJ are not proposed among the most frequent constituents by U-DOP while larger constituents like IN DT NN and DT JJ NN are in fact proposed.

### 3.2 Testing U-DOP on held-out sets and longer sentences (up to 40 words)

We were also interested in U-DOP's performance on a held-out test set such that we could compare the model with a *supervised* PCFG treebank grammar trained and tested on the same data (S-PCFG). We started by testing U-DOP on 10 different 90%/10% splits of the WSJ10, where 90% was used for inducing the trees, and 10% to parse new sentences by subtrees from the binary trees from the training set (or actually a PCFG-reduction thereof). The supervised PCFG was right-binarized as in Klein and Manning (2005). The following table shows the results.

| Model | UP | UR | F1 |
|---|---|---|---|
| U-DOP | 70.6 | **88.1** | 78.3 |
| S-PCFG | **84.0** | 79.8 | **81.8** |

Table 3. Average f-scores of U-DOP compared to a supervised PCFG (S-PCFG) on 10 different 90-10 splits of the WSJ10

Comparing table 1 with table 3, we see that on 10 held-out WSJ10 test sets U-DOP performs with an average f-score of 78.3% (SD=2.1%) only slightly worse than when using the entire WSJ10 corpus (78.5%). Next, note that U-DOP's results come near to the average performance of a binarized supervised PCFG which achieves 81.8% unlabeled f-score (SD=1.8%). U-DOP's unlabeled recall is even higher than that of the supervised PCFG. Moreover, according to paired *t*-testing, the differences in f-scores were *not* statistically significant. (If the PCFG was not post-binarized, its average f-score was 89.0%.)

As a final test case for this paper, we were interested in evaluating U-DOP on WSJ sentences ≤ 40 words, i.e. the WSJ40, which is with almost 50,000 sentences a much more challenging test case than the relatively small WSJ10. The main problem for U-DOP is the astronomically large number of possible binary trees for longer sentences, which therefore need to be even more heavily pruned than before.

We used a similar sampling heuristic as in section 2. We started by taking 100% of the trees for sentences ≤ 7 words. Next, for longer sentences we reduced this percentage with the relative increase of the Catalan number. This effectively means that we randomly selected the same number of trees for each sentence ≥ 8 words, which is 132 (i.e. the number of possible binary trees for a 7-word sentence). As mentioned in section 2, our sampling approach favors more frequent trees, and trees with more frequent subtrees. The binary tree-set obtained in this way for the WSJ40 consists of 5.11 * $10^6$ different trees. This resulted in a total of 88+ million distinct PCFG rules according to the reduction technique in section 2. As this is the largest PCFG we have ever attempted to parse with, it was prohibitive to estimate the most probable parse tree from 100 most probable derivations using Viterbi *n*-best. Instead, we used a beam of only 15 most probable derivations, and selected the most probable parse from these. (The number 15 is admittedly ad hoc, and was inspired by the performance of the so-called SL-DOP model in Bod 2002, 2003). The following table shows the results of U-DOP on the WSJ40 using 10 different 90-10 splits, compared to a supervised binarized PCFG (S-PCFG) and a supervised binarized DOP model (S-DOP) on the same data.

| Model | F1 |
|---|---|
| U-DOP | 64.2 |
| S-PCFG | 64.7 |
| S-DOP | 81.9 |

Table 4. Performance of U-DOP on WSJ40 using10 different 90-10 splits, compared to a binarized S-PCFG and a binarized S-DOP model.

Table 4 shows that U-DOP obtains about the same results as a binarized supervised PCFG on WSJ sentences ≤ 40 words. Moreover, the differences between U-DOP and S-PCFG were not statistically significant. This result is important as it shows that it is possible to parse the rather challinging WSJ in a completely *unsupervised* way obtaining roughly the same accuracy as a *supervised* PCFG. This seems to be in contrast with the CCM model which quickly degrades if sentence length is increased (see Klein 2005). As Klein (2005: 97) notes, CCM's strength is finding common short constituent chunks. U-DOP on the other hand has a preference for large (even largest possible) constituent chunks. Klein (2005: 97) reports that the combination of CCM and DMV seems to be more stable with increasing sentence length. It would be extremely interesting to see how DMV+CCM performs on the WSJ40.

It should be kept in mind that simple treebank PCFGs do not constitute state-of-the-art supervised parsers. Table 4 indicates that U-DOP's performance remains still far behind that of S-DOP (and indeed of other state-of-the-art supervised parsers such as Bod 2003 or Charniak and Johnson 2005). Moreover, if S-DOP is not post-binarized, its average f-score on the WSJ40 is 90.1% -- and there are some hybrid DOP models that obtain even higher scores (see Bod 2003). Our long-term goal is to try to outperform S-DOP by U-DOP. An important advantage of U-DOP is of course that it only needs unannotated data of which unlimited quanitities are available. Thus it would be interesting to test how U-DOP performs if trained on e.g. 100 times more data. Yet, as long as we compute our f-scores on *hand*-annotated data like Penn's WSJ, the S-DOP model is clearly at an advantage. We therefore plan to compare U-DOP and S-DOP (and other supervised parsers) in a concrete application such as phrase-based machine translation or as a language model for speech recognition.

## 4 Conclusions

We have shown that the general DOP approach can be generalized to unsupervised learning, effectively leading to a single model for both supervised and unsupervised parsing. Our new model, U-DOP, uses all subtrees from (in principle) all binary trees of a set of sentences to compute the most probable parse trees for (new) sentences. Although heavy pruning of trees is necessary to make our approach feasible in practice, we obtained competitive results on English, German and Chinese data. Our parsing results are similar to the performance of a binarized supervised PCFG on the WSJ ≤ 40 sentences. This triggers the provocative question as to whether it is possible to beat supervised parsing by unsupervised parsing. To cope with the problem of evaluation, we propose to test U-DOP in specific applications rather than on hand-annotated data.

## References

Bod, R. 1998. *Beyond Grammar: An Experience-Based Theory of Language*, Stanford: CSLI Publications (Lecture notes number 88), distributed by Cambridge University Press.

Bod, R. 2000. An improved parser for data-oriented lexical-functional analysis. *Proceedings ACL'2000*, Hong Kong.

Bod, R. 2002. A unified model of structural organization in language and music. *Journal of*

*Artificial Intelligence Research* 17(2002), 289-308.

Bod, R., R. Scha and K. Sima'an (eds.) 2003. *Data-Oriented Parsing*. CSLI Publications/University of Chicago Press.

Bod, R. 2003. An efficient implementation of a new DOP model. *Proceedings EACL'2003*, Budapest.

Bod, R. 2006. Exemplar-based syntax: How to get productivity from examples? *The Linguistic Review* 23(3), Special Isssue on Exemplar-Based Models in Linguistics.

Charniak, E. and M. Johnson 2005. Coarse-to-fine n-best parsing and Max-Ent discriminative reranking. *Proceedings ACL'2005*, Ann-Arbor.

Clark, A. 2000. Inducing syntactic categories by context distribution clustering. *Proceedings CONLL'2000.*

Clark, A. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. *Proceedings CONLL'2001*.

Collins, M. and N. Duffy 2002. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. *Proceedings ACL'2002*, Philadelphia.

Goodman, J. 2003. Efficient algorithms for the DOP model. In R. Bod, R. Scha and K. Sima'an (eds.). *Data-Oriented Parsing*, The University of Chicago Press.

Huang, L. and Chiang D. 2005. Better *k*-best parsing. *Proceedings IWPT'2005,* Vancouver.

Johnson, M. 2002. The DOP estimation method is biased and inconsistent. *Computational Linguistics* 28, 71-76.

Klein, D. 2005. *The Unsupervised Learning of Natural Language Structure*. PhD thesis, Stanford University.

Klein, D. and C. Manning 2002. A general constituent-context model for improved grammar induction. *Proceedings ACL'2002*, Philadelphia.

Klein, D. and C. Manning 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. *Proceedings ACL'2004*, Barcelona.

Klein, D. and C. Manning 2005. Natural language grammar induction with a generative constituent-context model. *Pattern Recognition* 38, 1407-1419.

Schütze, H. 1995. Distributional part-of-speech tagging. *Proceedings ACL'1995*, Dublin.

Sima'an, K. 1996. Computational complexity of probabilistic disambiguation by means of tree grammars. *Proceedings COLING'1996,* Copenhagen.

Skut, W., B. Krenn, T. Brants and H. Uszkoreit 1997. An annotation scheme for free word order languages. *Proceedings ANLP'97.*

Xue, N., F. Chiou and M. Palmer 2002. Building a large-scale annotated Chinese corpus. *Proceedings COLING 2002*, Taipei.

van Zaanen, M. 2000. ABL: Alignment-Based Learning. *Proceedings COLING'2000*, Saarbrücken.

Zollmann, A. and K. Sima'an 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, in press.

Zuidema, W. 2006. Theoretical evaluation of estimation methods for data-oriented parsing. *Proceedings EACL'2006*, Trento.