An All-Subtrees Approach to Unsupervised Parsing

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 investigate generalizations of the all-subtrees "DOP" approach to unsupervised parsing. Unsupervised DOP models assign all possible binary trees to a set of sentences and next use (a large random subset of) all subtrees from these binary trees to compute the most probable parse trees. We will test both a relative frequency estimator for unsupervised DOP and a maximum likelihood estimator which is known to be statistically consistent. We report state-of-the-art results on English (WSJ), German (NEGRA) and Chinese (CTB) data. To the best of our knowledge this is the first paper which tests a maximum likelihood estimator for DOP on the Wall Street Journal, leading to the surprising result that *an unsupervised parsing model beats a widely used supervised model (a treebank PCFG).*

1 Introduction

The problem of bootstrapping syntactic structure from unlabeled data has regained considerable interest. 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 steady progress in the field. Where van Zaanen (2000) achieved 39.2% unlabeled f-score on ATIS word strings, Clark (2001) reports 42.0% on the same data, and Klein and Manning (2002) obtain 51.2% f-score on ATIS part-of-speech strings using a constituent-context model called CCM. On Penn Wall Street Journal p-o-s-strings ≤ 10 (WSJ10), Klein and Manning (2002) report 71.1% unlabeled f-score with CCM. And the hybrid approach of Klein and Manning (2004), which combines constituency and dependency models, yields 77.6% f-score.

Bod (2006) shows that a further improvement on the WSJ10 can be achieved by an unsupervised generalization of the all-subtrees approach known as Data-Oriented Parsing (DOP). This unsupervised DOP model, coined U-DOP, first assigns all possible unlabeled binary trees to a set of sentences and next uses all subtrees from (a large subset of) these trees to compute the most probable parse trees. Bod (2006) reports that U-DOP not only outperforms previous unsupervised parsers but that its performance is as good as a binarized *supervised* parser (i.e. a treebank PCFG) on the WSJ.

A possible drawback of U-DOP, however, is the statistical inconsistency of its estimator (Johnson 2002) which is inherited from the DOP1 model (Bod 1998). That is, even with unlimited training data, U-DOP's estimator is not guaranteed to converge to the correct weight distribution. Johnson (2002: 76) argues in favor of a maximum likelihood estimator for DOP which *is* statistically consistent. As it happens, in Bod (2000) we already developed such a DOP model, termed ML-DOP, which reestimates the subtree probabilities by a maximum likelihood procedure based on Expectation-Maximization. Although cross-validation is needed to avoid overlearning, ML-DOP outperforms DOP1 on the OVIS corpus (Bod 2000).

This raises the question whether we can create an *unsupervised* DOP model which is also

statistically consistent. In this paper we will show that an unsupervised version of ML-DOP can be constructed along the lines of U-DOP. We will start out by summarizing DOP, U-DOP and ML-DOP, and next create a new unsupervised model called UML-DOP. We report that UML-DOP not only obtains higher parse accuracy than U-DOP on three different domains, but that it also achieves this with *fewer* subtrees than U-DOP. To the best of our knowledge, this paper presents the first *unsupervised* parser that outperforms a widely used *supervised* parser on the WSJ, i.e. a treebank PCFG. We will raise the question whether the end of supervised parsing is in sight.

2 DOP

The key idea of DOP is this: given an annotated corpus, use all subtrees, regardless of size, to parse new sentences. The DOP1 model in Bod (1998) computes the probabilities of parse trees and sentences from the relative frequencies of the subtrees. Although it is now known that DOP1's relative frequency estimator is statistically inconsistent (Johnson 2002), the model yields excellent empirical results and has been used in state-of-the-art systems. Let's illustrate DOP1 with a simple example. Assume a corpus consisting of only two trees, as given in figure 1.

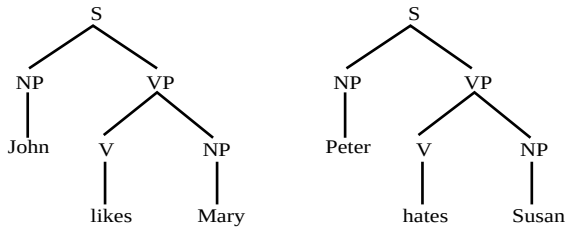

Figure 1. A corpus of two trees

New sentences may be derived by combining fragments, i.e. subtrees, from this corpus, by means of a node-substitution operation indicated as ∘. Node-substitution identifies the leftmost nonterminal frontier node of one subtree with the root node of a second subtree (i.e., the second subtree is *substituted* on the leftmost nonterminal frontier node of the first subtree). Thus a new sentence such as *Mary likes Susan* can be derived by combining subtrees from this corpus, shown in figure 2.

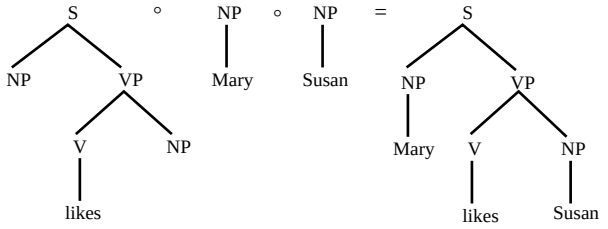

Figure 2. A derivation for *Mary likes Susan*

Other derivations may yield the same tree, e.g.:

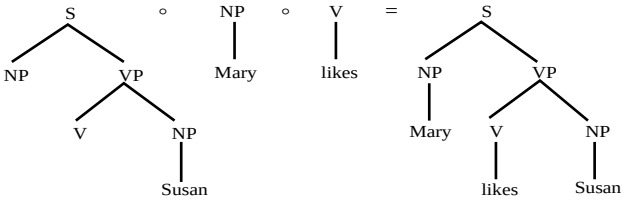

Figure 3. Another derivation yielding same tree

DOP1 computes the probability of a subtree t as the probability of selecting t among all corpus subtrees that can be substituted on the same node as t. This probability is computed as the number of occurrences of t in the corpus, $|\, t \,|$, divided by the total number of occurrences of all subtrees t' with the same root label as t.[1] Let $r(t)$ return the root label of t. Then we may write:

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

The probability of a derivation $t_1 \circ ... \circ t_n$ is computed by the product of the probabilities of its subtrees t_i:

$$P(t_1 \circ ... \circ t_n) = \prod_i P(t_i)$$

As we have seen, there may be several distinct derivations that generate the same parse tree. The probability of a parse tree T is the sum of the

[1] This subtree probability is redressed by a simple correction factor discussed in Goodman (2003: 136) and Bod (2003).

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

$$P(T) = \sum_d \prod_i P(t_{id})$$

Thus DOP1 considers counts of subtrees of a wide range of sizes: everything from counts of single-level rules to entire trees is taken into account to compute the most probable parse tree of a sentence. A disadvantage of the approach may be that an extremely large number of subtrees (and derivations) must be considered. Fortunately there exists a compact isomorphic PCFG-reduction of DOP1 whose size is linear rather than exponential in the size of the training set (Goodman 2003). Moreover, Collins and Duffy (2002) show how a tree kernel can be applied to DOP1's all-subtrees representation. The currently most successful version of DOP1 uses a PCFG-reduction of the model with an n-best parsing algorithm (Bod 2003).

3 U-DOP

U-DOP extends DOP1 to unsupervised parsing (Bod 2006). Its key idea is to assign all unlabeled binary trees to a set of sentences and to next use (in principle) all subtrees from these binary trees to parse new sentences. U-DOP thus proposes one of the richest possible models in bootstrapping trees. Previous models like Klein and Manning's (2002, 2005) CCM model limit the dependencies to "contiguous subsequences of a sentence". This means that CCM neglects dependencies that are *non*-contiguous such as between *more* and *than* in "*BA carried more people than cargo*". Instead, U-DOP's all-subtrees approach captures both contiguous and non-contiguous lexical dependencies.

As with most other unsupervised parsing models, U-DOP induces trees for p-o-s strings rather than for word strings. The extension to word strings is straightforward as there exist highly accurate unsupervised part-of-speech taggers (e.g. Schütze 1995) which can be directly combined with unsupervised parsers.

To give an illustration of U-DOP, consider the WSJ p-o-s string NNS VBD JJ NNS which may correspond for instance to the sentence *Investors suffered heavy losses*. U-DOP starts by assigning all possible binary trees to this string, where each root node is labeled *S* and each internal node is labeled *X*. Thus NNS VBD JJ NNS has a total of five binary trees shown in figure 4 -- where for readability we add words as well.

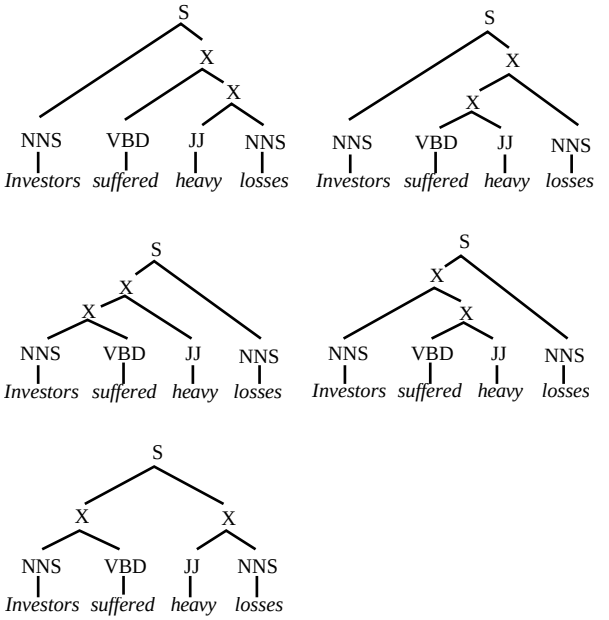

Figure 4. All binary trees for NNS VBD JJ NNS (*Investors suffered heavy losses*)

While we can efficiently represent the set of all binary trees of a string by means of a chart, we need to unpack the chart if we want to extract subtrees from this set of binary trees. And since the total number of binary trees for the small WSJ10 is almost 12 million, it is doubtful whether we can apply the unrestricted U-DOP model to such a corpus. U-DOP therefore randomly samples a large subset from the total number of parse trees from the chart (see Bod 2006) and next converts the subtrees from these parse trees into a PCFG-reduction (Goodman 2003). Since the computation of the most probable parse tree is NP-complete (Sima'an 1996), U-DOP estimates the most probable tree from the 100 most probable derivations using Viterbi n-best parsing. We could also have used the more efficient k-best hypergraph parsing technique by Huang and Chiang (2005), but we have not yet incorporated this into our implementation.

To give an example of the dependencies that U-DOP can take into account, consider the following subtrees in figure 5 from the trees in

figure 4 (where we again add words for readability). These subtrees show that U-DOP takes into account both contiguous and non-contiguous substrings.

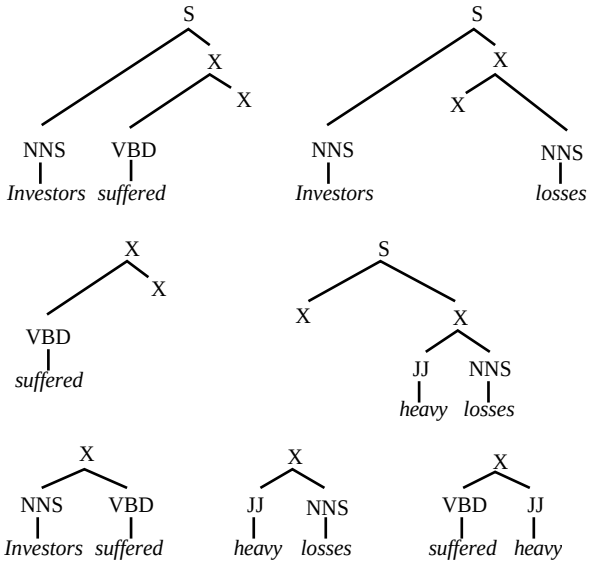

Figure 5. Some subtrees from trees in figure 4

Of course, if we only had the sentence *Investors suffered heavy losses* in our corpus, there would be no difference in probability between the five parse trees in figure 4. However, if we also 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.

4 ML-DOP

ML-DOP (Bod 2000) extends DOP with a maximum likelihood reestimation technique based on the expectation-maximization (EM) algorithm (Dempster et al. 1977) which is known to be statistically consistent (Shao 1999). ML-DOP reestimates DOP's subtree probabilities in an iterative way until the changes become negligible. The following exposition of ML-DOP is heavily based on previous work by Bod (2000) and Magerman (1993).

It is important to realize that there is an implicit assumption in DOP that all possible derivations of a parse tree contribute equally to the total probability of the parse tree. This is equivalent to saying that there is a hidden component to the model, and that DOP can be trained using an EM algorithm to determine the maximum likelihood estimate for the training data. The EM algorithm for this ML-DOP model is related to the Inside-Outside algorithm for context-free grammars, but the reestimation formula is complicated by the presence of subtrees of depth greater than 1. To derive the reestimation formula, it is useful to consider the state space of all possible derivations of a tree.

The derivations of a parse tree T can be viewed as a state trellis, where each state contains a partially constructed tree in the course of a leftmost derivation of T. s_t denotes a state containing the tree t which is a subtree of T. The state trellis is defined as follows.

The initial state, s_0, is a tree with depth zero, consisting of simply a root node labeled with S. The final state, s_T, is the given parse tree T.

A state s_t is connected forward to all states s_{tf} such that $t_f = t \circ t'$, for some t'. Here the appropriate t' is defined to be $t_f - t$.

A state s_t is connected backward to all states s_{tb} such that $t = t_b \circ t'$, for some t'. Again, t' is defined to be $t - t_b$.

The construction of the state lattice and assignment of transition probabilities according to the ML-DOP model is called the forward pass. The probability of a given state, $P(s)$, is referred to as $\alpha(s)$. The forward probability of a state s_t is computed recursively

$$\alpha(s_t) = \sum_{s_{t_b}} \alpha(s_{t_b})\, P(t - t_b).$$

The backward probability of a state, referred to as $\beta(s)$, is calculated according to the following recursive formula:

$$\beta(s_t) = \sum_{s_{t_f}} \beta(s_{t_f})\, P(t_f - t)$$

where the backward probability of the goal state is set equal to the forward probability of the goal state, $\beta(s_T) = \alpha(s_T)$.

The update formula for the count of a subtree t is (where $r(t)$ is the root label of t):

$$ct(t) = \sum_{s_{t_b}: \exists s_{t_f}, t_b \circ t = t_f} \frac{\beta(s_{t_f})\alpha(s_{t_b})P(t \mid r(t))}{\alpha(s_{goal})}$$

The updated probability distribution, $P'(t \mid r(t))$, is defined to be

$$P'(t \mid r(t)) = \frac{ct(t)}{ct(r(t))}$$

where $ct(r(t))$ is defined as

$$ct(r(t)) = \sum_{t':\ r(t')=r(t)} ct(t')$$

In practice, ML-DOP starts out by assigning the same relative frequencies to the subtrees as DOP1, which are next reestimated by the formulas above. We may in principle start out with any initial parameters, including random initializations, but since ML estimation is known to be very sensitive to the initialization of the parameters, it is convenient to start with parameters that are known to perform well.

To avoid overtraining, ML-DOP uses the subtrees from one half of the training set to be trained on the other half, and vice versa. This cross-training is important since otherwise UML-DOP would assign the training set trees their empirical frequencies and assign zero weight to all other subtrees (cf. Prescher et al. 2004). The updated probabilities are iteratively reestimated until the decrease in cross-entropy becomes negligible. Unfortunately, no compact PCFG-reduction of ML-DOP is known. As a consequence, parsing with ML-DOP is very costly and the model has hitherto never been tested on corpora larger than OVIS (Bonnema et al. 1997). Yet, we will show that by clever pruning we can extend our experiments not only to the WSJ, but also to the German NEGRA and the Chinese CTB. (Zollmann and Sima'an 2005 propose a different consistent estimator for DOP, which we cannot go into here).

5 UML-DOP

Analogous to U-DOP, UML-DOP is an unsupervised generalization of ML-DOP: it first assigns all unlabeled binary trees to a set of sentences and next extracts a large (random) set of subtrees from this tree set. It then reestimates the initial probabilities of these subtrees by ML-DOP on the sentences from a held-out part of the tree set. The training is carried out by dividing the tree set into two equal parts, and reestimating the subtrees from one part on the other. As initial probabilities we use the subtrees' relative frequencies as described in section 2 (smoothed by Good-Turing -- see Bod 1998), though it would also be interesting to see how the model works with other initial parameters, in particular with the usage frequencies proposed by Zuidema (2006).

As with U-DOP, the total number of subtrees that can be extracted from the binary tree set is too large to be fully taken into account. Together with the high computational cost of reestimation we propose even more drastic pruning than we did in Bod (2006) for U-DOP. That is, while for sentences ≤ 7 words we use all binary trees, for each sentence ≥ 8 words we randomly sample a fixed number of 128 trees (which effectively favors more frequent trees). The resulting set of trees is referred to as the binary tree set. Next, we randomly extract for each subtree-depth a fixed number of subtrees, where the depth of subtree is the longest path from root to any leaf. This has roughly the same effect as the correction factor used in Bod (2003, 2006). That is, for each particular depth we sample subtrees by first randomly selecting a node in a random tree from the binary tree set after which we select random expansions from that node until a subtree of the particular depth is obtained. For our experiments in section 6, we repeated this procedure 200,000 times for each depth. The resulting subtrees are then given to ML-DOP's reestimation procedure. Finally, the reestimated subtrees are used to compute the most probable parse trees for all sentences using Viterbi *n*-best, as described in section 3, where the most probable parse is estimated from the 100 most probable derivations.

A potential criticism of (U)ML-DOP is that since we use DOP1's relative frequencies as initial parameters, ML-DOP may only find a local maximum nearest to DOP1's estimator. But this is of course a criticism against any iterative ML approach: it is not guaranteed that the global maximum is found (cf. Manning and Schütze 1999: 401). Nevertheless we will see that our reestimation

procedure leads to significantly better accuracy compared to U-DOP (the latter would be equal to UML-DOP under 0 iterations). Moreover, in contrast to U-DOP, UML-DOP *can* be theoretically motivated: it maximizes the likelihood of the data using the statistically consistent EM algorithm.

6 Experiments: Can we beat supervised by unsupervised parsing?

To compare UML-DOP to U-DOP, we started out with the WSJ10 corpus, which contains 7422 sentences ≤ 10 words after removing empty elements and punctuation. We used the same evaluation metrics for unlabeled precision (UP) and unlabeled recall (UR) as defined in Klein (2005: 21-22). Klein's definitions differ slightly 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 unlabeled f-score F1 which is defined as the harmonic mean of UP and UR: F1 = 2*UP*UR/(UP+UR).

For the WSJ10, we obtained a binary tree set of $5.68 * 10^5$ trees, by extracting the binary trees as described in section 5. From this binary tree set we sampled 200,000 subtrees for each subtree-depth. This resulted in a total set of roughly $1.7 * 10^6$ subtrees that were reestimated by our maximum-likelihood procedure. The decrease in cross-entropy became negligible after 14 iterations (for both halfs of WSJ10). After computing the most probable parse trees, UML-DOP achieved an f-score of 82.9% which is a 20.5% error reduction compared to U-DOP's f-score of 78.5% on the same data (Bod 2006).

We next tested UML-DOP on two additional domains which were also used in Klein and Manning (2004) and Bod (2006): the German NEGRA10 (Skut et al. 1997) and the Chinese CTB10 (Xue et al. 2002) both containing 2200+ sentences ≤ 10 words after removing punctuation. Table 1 shows the results of UML-DOP compared to U-DOP, the CCM model by Klein and Manning (2002), the DMV dependency learning model by Klein and Manning (2004) as well as their combined model DMV+CCM.

Table 1 shows that UML-DOP scores better than U-DOP and Klein and Manning's models in all cases. It thus pays off to not only use subtrees rather than substrings (as in CCM) but to also reestimate the subtrees' probabilities by a maximum-likelihood procedure rather than using their (smoothed) relative frequencies (as in U-DOP). Note that UML-DOP achieves these improved results with *fewer* subtrees than U-DOP, due to UML-DOP's more drastic pruning of subtrees. It is also noteworthy that UML-DOP, like U-DOP, does not employ a separate class for non-constituents, so-called distituents, while CCM and CCM+DMV do. (Interestingly, the top 10 most frequently learned constituents by UML-DOP were exactly the same as by U-DOP -- see the relevant table in Bod 2006).

Model	English (WSJ10)	German (NEGRA10)	Chinese (CTB10)
CCM	71.9	61.6	45.0
DMV	52.1	49.5	46.7
DMV+CCM	77.6	63.9	43.3
U-DOP	78.5	65.4	46.6
UML-DOP	**82.9**	**67.0**	**47.2**

Table 1. F-scores of UML-DOP compared to previous models on the same data

We were also interested in testing UML-DOP on longer sentences. We therefore included all WSJ sentences up to 40 words after removing empty elements and punctuation (WSJ40) and again sampled 200,000 subtrees for each depth, using the same method as before. Furthermore, we compared UML-DOP against a supervised binarized PCFG, i.e. a treebank PCFG whose simple relative frequency estimator corresponds to maximum likelihood (Chi and Geman 1998), and which we shall refer to as "ML-PCFG". To this end, we used a random 90%/10% division of WSJ40 into a training set and a test set. The ML-PCFG had thus access to the Penn WSJ trees in the training set, while UML-DOP had to bootstrap all structure from the *flat* strings from the training set to next parse the 10% test set -- clearly a much more challenging task. Table 2 gives the results in terms of f-scores.

The table shows that UML-DOP scores better than U-DOP, also for WSJ40. Our results on WSJ10 are somewhat lower than in table 1 due to the use of a smaller training set of 90% of the data. But the most surprising result is that UML-DOP's f-

score is higher than the *supervised* binarized treebank PCFG (ML-PCFG) for both WSJ10 and WSJ40. In order to check whether this difference is statistically significant, we additionally tested on 10 different 90/10 divisions of the WSJ40 (which were the same splits as in Bod 2006). For these splits, UML-DOP achieved an average f-score of 66.9%, while ML-PCFG obtained an average f-score of 64.7%. The difference in accuracy between UML-DOP and ML-PCFG was statistically significant according to paired *t*-testing ($p \leq 0.05$). To the best of our knowledge this means that we have shown for the first time that *an unsupervised parsing model (UML-DOP) outperforms a widely used supervised parsing model (a treebank PCFG) on the WSJ40*.

Model	WSJ10	WSJ40
U-DOP	78.1	63.9
UML-DOP	**82.5**	**66.4**
ML-PCFG	81.5	64.6

Table 2. F-scores of U-DOP, UML-DOP and a supervised treebank PCFG (ML-PCFG) for a random 90/10 split of WSJ10 and WSJ40.

We should keep in mind that (1) a treebank PCFG is not state-of-the-art: its performance is mediocre compared to e.g. Bod (2003) or McClosky et al. (2006), and (2) that our treebank PCFG is binarized as in Klein and Manning (2005) to make results comparable. To be sure, the unbinarized version of the treebank PCFG obtains 89.0% average f-score on WSJ10 and 72.3% average f-score on WSJ40. Remember that the Penn Treebank annotations are often exceedingly flat, and many branches have arity larger than two. It would be interesting to see how UML-DOP performs if we also accept ternary (and wider) branches -- though the total number of possible trees that can be assigned to strings would then further explode.

UML-DOP's performance still remains behind that of *supervised* (binarized) DOP parsers, such as DOP1, which achieved 81.9% average f-score on the 10 WSJ40 splits, and ML-DOP, which performed slightly better with 82.1% average f-score. And if DOP1 and ML-DOP are not binarized, their average f-scores are respectively 90.1% and 90.5% on WSJ40. However, DOP1 and ML-DOP heavily depend on annotated data whereas UML-DOP only needs unannotated data. It would thus be interesting to see how close UML-DOP can get to ML-DOP's performance if we enlarge the amount of training data.

7 Conclusion: Is the end of supervised parsing in sight?

Now that we have outperformed a well-known supervised parser by an unsupervised one, we may raise the question as to whether the end of supervised NLP comes in sight. All supervised parsers are reaching an asymptote and further improvement does not seem to come from more hand-annotated data but by adding unsupervised or semi-unsupervised techniques (cf. McClosky et al. 2006). Thus if we modify our question as: does the *exclusively* supervised approach to parsing come to an end, we believe that the answer is certainly yes.

Yet we should neither rule out the possibility that entirely unsupervised methods will in fact surpass semi-supervised methods. The main problem is how to quantitatively compare these different parsers, as any evaluation on hand-annotated data (like the Penn treebank) will unreasonably favor semi-supervised parsers. There is thus is a quest for designing an annotation-independent evaluation scheme. Since parsers are becoming increasingly important in applications like syntax-based machine translation and structural language models for speech recognition, one way to go would be to compare these different parsing methods by isolating their contribution in improving a concrete NLP system, rather than by testing them against gold standard annotations which are inherently theory-dependent.

The initially disappointing results of inducing trees entirely from raw text was not so much due to the difficulty of the bootstrapping problem *per se*, but to (1) the poverty of the initial models and (2) the difficulty of finding theory-independent evaluation criteria. The time has come to fully reappraise unsupervised parsing models which should be trained on massive amounts of data, and be evaluated in a concrete application.

There is a final question as to how far the DOP approach to unsupervised parsing can be stretched. In principle we can assign all possible syntactic categories, semantic roles, argument

structures etc. to a set of given sentences and let the statistics decide which assignments are most useful in parsing new sentences. Whether such a massively maximalist approach is feasible can only be answered by empirical investigation in due time.

Acknowledgements

Thanks to Willem Zuidema, David Tugwell and especially to three anonymous reviewers whose unanymous suggestions on DOP and EM considerably improved the original paper. A substantial part of this research was carried out in the context of the NWO Exact project "Unsupervised Stochastic Grammar Induction from Unlabeled Data", project number 612.066.405.

References

Bod, R. 1998. *Beyond Grammar: An Experience-Based Theory of Language*, CSLI Publications, distributed by Cambridge University Press.

Bod, R. 2000. Combining semantic and syntactic structure for language modeling. *Proceedings ICSLP 2000*, Beijing.

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

Bod, R. 2006. Unsupervised Parsing with U-DOP. *Proceedings CONLL 2006*, New York.

Bonnema, R., R. Bod and R. Scha, 1997. A DOP model for semantic interpretation, *Proceedings ACL/EACL 1997*, Madrid.

Chi, Z. and S. Geman 1998. Estimation of Probabilistic Context-Free Grammars. *Computational Linguistics* 24, 299-305.

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.

Dempster, A., N. Laird and D. Rubin, 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society* 39, 1-38.

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

Huang, L. and D. Chiang 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.

Magerman, D. 1993. *Expectation-Maximization for Data-Oriented Parsing*, IBM Technical Report, Yorktown Heights, NY.

McClosky, D., E. Charniak and M. Johnson 2006. Effective self-training for parsing. *Proceedings HLT-NAACL 2006*, New York.

Manning, C. and H. Schütze 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Prescher, D., R. Scha, K. Sima'an and A. Zollmann 2004. On the statistical consistency of DOP estimators. *Proceedings CLIN 2004*, Leiden.

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

Shao, J. 1999. *Mathematical Statistics*. Springer Verlag, New York.

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 1997.*

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. What are the productive units of natural language grammar? A DOP approach to the automatic identification of constructions. *Proceedings CONLL 2006*, New York.