Statistical parsing with an automatically-extracted tree adjoining grammar

David Chiang

Department of Computer and Information Science
University of Pennsylvania
200 S 33rd St
Philadelphia PA 19104
dchiang@linc.cis.upenn.edu

Abstract

We discuss the advantages of lexicalized tree-adjoining grammar as an alternative to lexicalized PCFG for statistical parsing, describing the induction of a probabilistic LTAG model from the Penn Treebank and evaluating its parsing performance. We find that this induction method is an improvement over the EM-based method of (Hwa, 1998), and that the induced model yields results comparable to lexicalized PCFG.

1 Introduction

Why use tree-adjoining grammar for statistical parsing? Given that statistical natural language processing is concerned with the probable rather than the possible, it is not because TAG can describe constructions like arbitrarily large Dutch verb clusters. Rather, what makes TAG useful for statistical parsing are the *structural descriptions* it assigns to bread-and-butter sentences.

The approach of Chelba and Jelinek (1998) to language modeling is illustrative: even though the probability estimate of w appearing as the kth word can be conditioned on the entire history $w_1, \ldots, w_{k-1}$, the quantity of available training data limits the usable context to about two words—but which two? A trigram model chooses w_{k-1} and w_{k-2} and works quite well; a model which chose w_{k-7} and w_{k-11} would probably work less well. But (Chelba and Jelinek, 1998) chooses the lexical heads of the two previous constituents as determined by a shift-reduce parser, and works better than a trigram model. Thus the (virtual) grammar serves to *structure* the history so that the two most useful words can be chosen, even though the structure of the problem itself is entirely linear.

Similarly, nothing about the parsing problem requires that we construct any structure other than phrase structure. But beginning with (Magerman, 1995) statistical parsers have used bilexical dependencies with great success. Since these dependencies are not encoded in plain phrase-structure trees, the standard approach has been to let the lexical heads percolate up the tree, so that when one lexical head is immediately dominated by another, it is understood to be dependent on it. Effectively, a dependency structure is made parasitic on the phrase structure so that they can be generated together by a context-free model.

However, this solution is not ideal. Aside from cases where context-free derivations are incapable of encoding both constituency and dependency (which are somewhat isolated and not of great interest for statistical parsing) there are common cases where percolation of single heads is not sufficient to encode dependencies correctly—for example, relative clause attachment or raising/auxiliary verbs (see Section 3). More complicated grammar transformations are necessary.

A more suitable approach is to employ a grammar formalism which produces structural descriptions that can encode both constituency and dependency. Lexicalized TAG is such a formalism, because it assigns to each sentence not only a parse tree, which is built out of elementary trees and is interpreted as encoding constituency, but a *derivation tree*, which records how the various elementary trees were combined together and is commonly intepreted as encoding dependency. The ability of probabilistic LTAG to
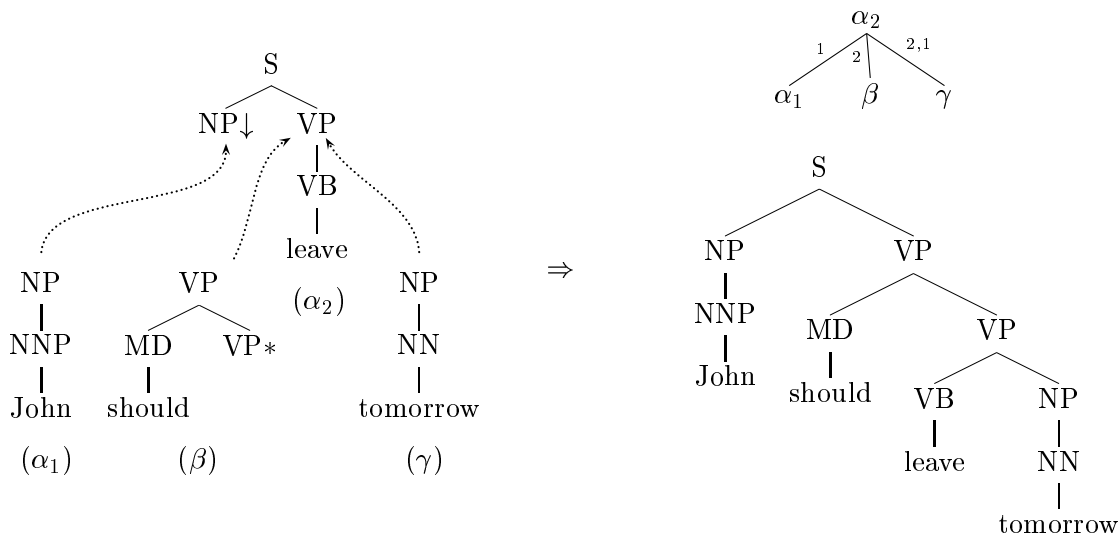
Figure 1: Grammar and derivation for "John should leave tomorrow."

model bilexical dependencies was noted early on by (Resnik, 1992).

It turns out that there are other pieces of contextual information that need to be explicitly accounted for in a CFG by grammar transformations but come for free in a TAG. We discuss a few such cases in Section 3. In Sections 4 and 5 we describe an experiment to test the parsing accuracy of a probabilistic TAG extracted automatically from the Penn Treebank. We find that the automatically-extracted grammar gives an improvement over the EM-based induction method of (Hwa, 1998), and that the parser performs comparably to lexicalized PCFG parsers, though certainly with room for improvement.

We emphasize that TAG is attractive not because it can do things that CFG cannot, but because it does everything that CFG can, only more cleanly. (This is where the analogy with (Chelba and Jelinek, 1998) breaks down.) Thus certain possibilities which were not apparent in a PCFG framework or prohibitively complicated might become simple to implement in a PTAG framework; we conclude by offering two such possibilities.

2 The formalism

The formalism we use is a variant of lexicalized tree-insertion grammar (LTIG), which is in turn a restriction of LTAG (Schabes and Waters, 1995). In this variant there are three kinds of elementary tree: initial, (predicative) auxiliary, and modifier, and three composition operations: substitution, adjunction, and sister-adjunction.

Auxiliary trees and adjunction are restricted as in TIG: essentially, no wrapping adjunction or anything equivalent to wrapping adjunction is allowed. Sister-adjunction is not an operation found in standard definitions of TAG, but is borrowed from D-Tree Grammar (Rambow et al., 1995). In sister-adjunction the root of a modifier tree is added as a new daughter to any other node. (Note that as it stands sister-adjunction is completely unconstrained; it will be constrained by the probability model.) We introduce this operation simply so we can derive the flat structures found in the Penn Treebank. Following (Schabes and Shieber, 1994), multiple modifier trees can be sister-adjoined at a single site, but only one auxiliary tree may be adjoined at a single node.

Figure 1 shows an example grammar and the derivation of the sentence "John should leave tomorrow." The derivation tree encodes this process, with each arc corresponding to a composition operation. Arcs corresponding to substitution and adjunction are labeled with the Gorn address[1] of the substitution or ad-

---

[1] A Gorn address is a list of integers: the root of a tree has address ϵ, and the jth child of the node with

junction site. An arc corresponding to the sister-adjunction of a tree between the ith and $i + 1$th children of η (allowing for two imaginary children beyond the leftmost and rightmost children) is labeled η, i.

This grammar, as well as the grammar used by the parser, is lexicalized in the sense that every elementary tree has exactly one terminal node, its lexical *anchor*.

Since sister-adjunction can be simulated by ordinary adjunction, this variant is, like TIG (and CFG), weakly context-free and $O(n^3)$-time parsable. Rather than coin a new acronym for this particular variant, we will simply refer to it as "TAG" and trust that no confusion will arise.

The parameters of a probabilistic TAG (Resnik, 1992; Schabes, 1992) are:

$$\sum_{\alpha} P_i(\alpha) = 1$$
$$\sum_{\alpha} P_s(\alpha \mid \eta) = 1$$
$$\sum_{\beta} P_a(\beta \mid \eta) + P_a(NONE \mid \eta) = 1$$

where α ranges over initial trees, β over auxiliary trees, γ over modifier trees, and η over nodes. $P_i(\alpha)$ is the probability of beginning a derivation with α; $P_s(\alpha \mid \eta)$ is the probability of substituting α at η; $P_a(\beta \mid \eta)$ is the probability of adjoining β at η; finally, $P_a(NONE \mid \eta)$ is the probability of nothing adjoining at η. (Carroll and Weir, 1997) suggest other parameterizations worth exploring as well.

Our variant adds another set of parameters:

$$\sum_{\gamma} P_{sa}(\gamma \mid \eta, i, f) + P_{sa}(STOP \mid \eta, i, f) = 1$$

This is the probability of sister-adjoining γ between the ith and $i + 1$th children of η (as before, allowing for two imaginary children beyond the leftmost and rightmost children). Since multiple modifier trees can adjoin at the same location, $P_{sa}(\gamma)$ is also conditioned on a flag f which indicates whether γ is the first modifier tree (i.e., the one closest to the head) to adjoin at that location.

The probability of a derivation can then be expressed as a product of the probabilities of

---

address i has address $i \cdot j$.

the individual operations of the derivation. Thus the probability of the example derivation of Figure 1 would be

$$
\begin{aligned}
&P_i(\alpha_2) \cdot P_a(NONE \mid \alpha_2(\epsilon)) \cdot \\
&\quad P_s(\alpha_1 \mid \alpha_2(1)) \cdot P_a(\beta \mid \alpha_2(2)) \cdot \\
&\quad P_{sa}(\gamma \mid \alpha_2(2), 1, true) \cdot \\
&\quad P_{sa}(STOP \mid \alpha_2(2), 1, false) \cdot \\
&\quad P_{sa}(STOP \mid \alpha_2(\epsilon), 0, true) \cdot \ldots
\end{aligned}
$$

where $\alpha(i)$ is the node of α with address i.

We want to obtain a maximum-likelihood estimate of these parameters, but cannot estimate them directly from the Treebank, because the sample space of PTAG is the space of TAG derivations, not the derived trees that are found in the Treebank. One approach, taken in (Hwa, 1998), is to choose some grammar general enough to parse the whole corpus and obtain a maximum-likelihood estimate by EM. Another approach, taken in (Magerman, 1995) and others for lexicalized PCFGs and (Neumann, 1998; Xia, 1999; Chen and Vijay-Shanker, 2000) for LTAGs, is to use heuristics to reconstruct the derivations, and directly estimate the PTAG parameters from the reconstructed derivations. We take this approach as well. (One could imagine combining the two approaches, using heuristics to extract a grammar but EM to estimate its parameters.)

3 Some properties of probabilistic TAG

In a lexicalized TAG, because each composition brings together two lexical items, every composition probability involves a bilexical dependency. Given a CFG and head-percolation scheme, an equivalent TAG can be constructed whose derivations mirror the dependency analysis implicit in the head-percolation scheme.

Furthermore, there are some dependency analyses encodable by TAGs that are not encodable by a simple head-percolation scheme. For example, for the sentence "John should have left," Magerman's rules make *should* and *have* the heads of their respective VPs, so that there is no dependency between *left* and its subject *John* (see Figure 2a). Since nearly a quarter of nonempty subjects appear in such a configuration, this is not a small problem.
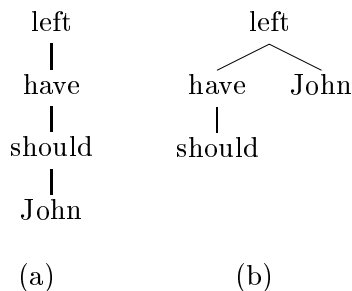
```
    left              left
     |               /    \
    have         have      John
     |               |
   should         should
     |
    John

     (a)              (b)
```

Figure 2: Bilexical dependencies for "John should have left."

(We could make VP the head of VP instead, but this would generate auxiliaries independently of each other, so that, for example, $P(\text{John leave}) > 0$.)

TAG can produce the desired dependencies (b) easily, using the grammar of Figure 1. A more complex lexicalization scheme for CFG could as well (one which kept track of two heads at a time, for example), but the TAG account is simpler and cleaner.

Bilexical dependencies are not the only nonlocal dependencies that can be used to improve parsing accuracy. For example, the attachment of an S depends on the presence or absence of the embedded subject (Collins, 1999); Treebank-style two-level NPs are mismodeled by PCFG (Collins, 1999; Johnson, 1998); the generation of a node depends on the label of its grandparent (Charniak, 2000; Johnson, 1998). In order to capture such dependencies in a PCFG-based model, they must be localized either by transforming the data or modifying the parser. Such changes are not always obvious *a priori* and often must be devised anew for each language or each corpus.

But none of these cases really requires special treatment in a PTAG model, because each composition probability involves not only a bilexical dependency but a "biarboreal" (tree-tree) dependency. That is, PTAG generates an entire elementary tree at once, conditioned on the entire elementary tree being modified. Thus dependencies that have to be stipulated in a PCFG by tree transformations or parser modifications are captured for free in a PTAG model. Of course, the price

that the PTAG model pays is sparser data; the backoff model must therefore be chosen carefully.

4 Inducing a stochastic grammar from the Treebank

4.1 Reconstructing derivations

We want to extract from the Penn Treebank an LTAG whose derivations mirror the dependency analysis implicit in the head-percolation rules of (Magerman, 1995; Collins, 1997). For each node η, these rules classify exactly one child of η as a head and the rest as either arguments or adjuncts. Using this classification we can construct a TAG derivation (including elementary trees) from a derived tree as follows:

1. If η is an adjunct, excise the subtree rooted at η to form a modifier tree.

2. If η is an argument, excise the subtree rooted at η to form an initial tree, leaving behind a substitution node.

3. If η has a right corner θ which is an argument with the same label as η (and all intervening nodes are heads), excise the segment from η down to θ to form an auxiliary tree.

Rules (1) and (2) produce the desired result; rule (3) changes the analysis somewhat by making subtrees with recursive arguments into predicative auxiliary trees. It produces, among other things, the analysis of auxiliary verbs described in the previous section. It is applied in a greedy fashion, with potential ηs considered top-down and potential θs bottom-up. The complicated restrictions on θ are simply to ensure that a well-formed TIG derivation is produced.

4.2 Parameter estimation and smoothing

Now that we have augmented the training data to include TAG derivations, we could try to directly estimate the parameters of the model from Section 2. But since the number of (tree, site) pairs is very high, the data would be too sparse. We therefore generate an elementary tree in two steps: first the tree template (that is, the elementary tree minus its
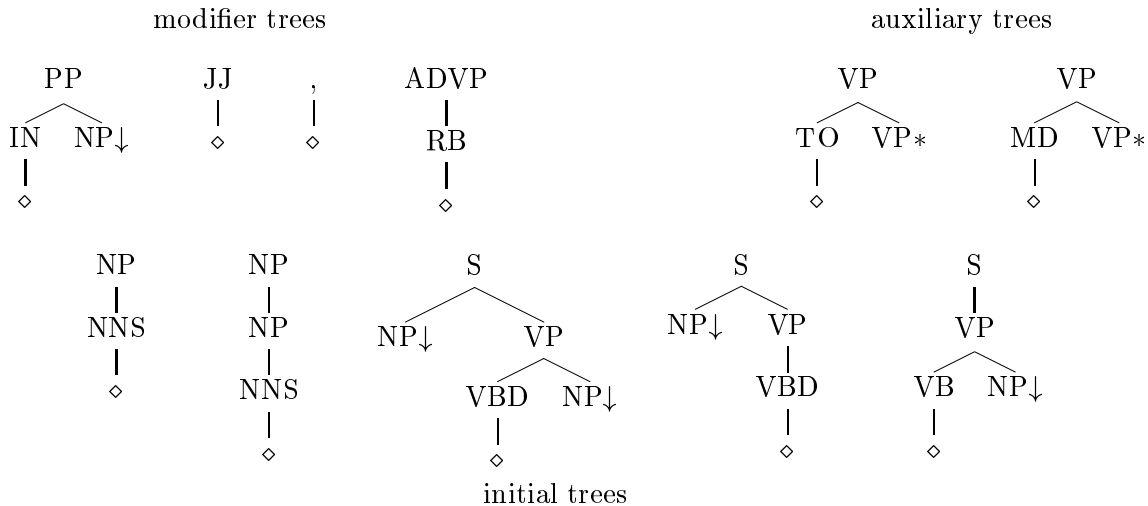
modifier trees

auxiliary trees

initial trees

Figure 3: A few of the more frequently-occurring tree templates. ◇ marks where the lexical anchor is inserted.

anchor), then the anchor. The probabilities are decomposed as follows:

$$
\begin{aligned}
P_i(\alpha) &= P_{i_1}(\tau_\alpha) P_{i_2}(w_\alpha \mid \tau_\alpha) \\
P_s(\alpha \mid \eta) &= P_{s_1}(\tau_\alpha \mid \eta)\cdot \\
&\quad P_{s_2}(w_\alpha \mid \tau_\alpha, t_\eta, w_\eta) \\
P_a(\beta \mid \eta) &= P_{a_1}(\tau_\beta \mid \eta)\cdot \\
&\quad P_{a_2}(w_\beta \mid \tau_\beta, t_\eta, w_\eta) \\
P_{sa}(\gamma \mid \eta, i, f) &= P_{sa_1}(\tau_\gamma \mid \eta, i, f)\cdot \\
&\quad P_{sa_2}(w_\gamma \mid \tau_\gamma, t_\eta, w_\eta, f)
\end{aligned}
$$

where τ_α is the tree template of α, t_α is the part-of-speech tag of the anchor, and w_α is the anchor itself.

The generation of the tree template has two backoff levels: at the first level, the anchor of η is ignored, and at the second level, the POS tag of the anchor as well as the flag f are ignored. The generation of the anchor has three backoff levels: the first two are as before, and the third just conditions the anchor on its POS tag. The backed-off models are combined by linear interpolation, with the weights chosen as in (Bikel et al., 1997).

5 The experiment

5.1 Extracting the grammar

We ran the algorithm given in Section 4.1 on sections 02–21 of the Penn Treebank. The extracted grammar is large (about 73,000 trees, with words seen fewer than four times replaced with the symbol *UNKNOWN*), but if we
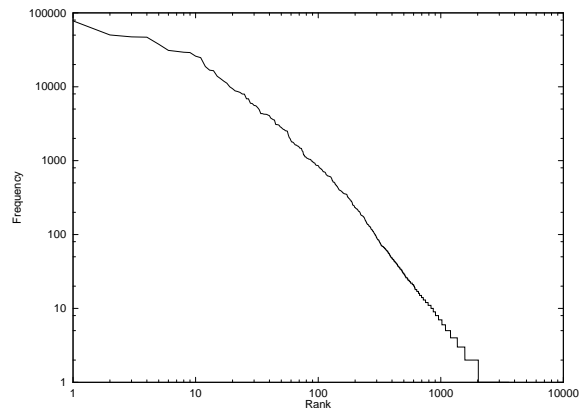

Figure 4: Frequency of tree templates versus rank (log-log)

consider elementary tree templates, the grammar is quite manageable: 3626 tree templates, of which 2039 occur more than once (see Figure 4).

The 616 most frequent tree-template types account for 99% of tree-template tokens in the training data. Removing all but these trees from the grammar increased the error rate by about 5% (testing on a subset of section 00). A few of the most frequent tree-templates are shown in Figure 3.

So the extracted grammar is fairly compact, but how complete is it? If we plot the growth of the grammar during training (Figure 5), it's not clear the grammar will ever converge, even though the very idea of a
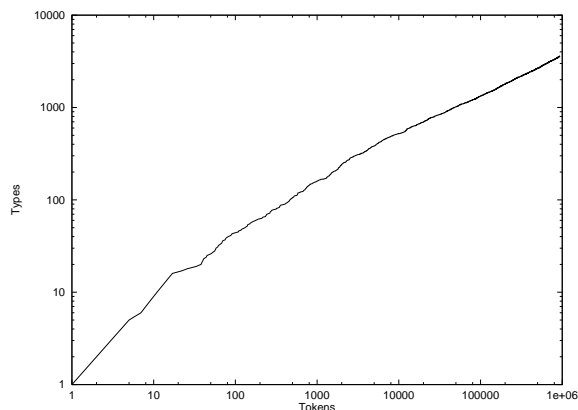
Figure 5: Growth of grammar during training (log-log)

grammar requires it. Three possible explanations are:

- New constructions continue to appear.

- Old constructions continue to be (erroneously) annotated in new ways.

- Old constructions continue to be combined in new ways, and the extraction heuristics fail to factor this variation out.

In a random sample of 100 once-seen elementary tree templates, we found (by casual inspection) that 34 resulted from annotation errors, 50 from deficiencies in the heuristics, and four apparently from performance errors. Only twelve appeared to be genuine.

Therefore the continued growth of the grammar is not as rapid as Figure 5 might indicate. Moreover, our extraction heuristics evidently have room to improve. The majority of trees resulting from deficiencies in the heuristics involved complicated coordination structures, which is not surprising, since coordination has always been problematic for TAG.

To see what the impact of this failure to converge is, we ran the grammar extractor on some held-out data (section 00). Out of 45082 tree tokens, 107 tree templates, or 0.2%, had not been seen in training. This amounts to about one unseen tree template every 20 sentences. When we consider lexicalized trees, this figure of course rises: out of the same 45082 tree tokens, 1828 lexicalized trees, or 4%, had not been seen in training.

So the coverage of the grammar is quite good. Note that even in cases where the parser encounters a sentence for which the (fallible) extraction heuristics would have produced an unseen tree template, it is possible that the parser will use other trees to produce the correct bracketing.

5.2 Parsing with the grammar

We used a CKY-style parser similar to the one described in (Schabes and Waters, 1996), with a modification to ensure completeness (because foot nodes are treated as empty, which CKY prohibits) and another to reduce useless substitutions. We also extended the parser to simulate sister-adjunction as regular adjunction and compute the flag f which distinguishes the first modifier from subsequent modifiers.

We use a beam search, computing the score of an item $[\eta, i, j]$ by multiplying it by the prior probability $P(\eta)$ (Goodman, 1997); any item with score less than 10^{-5} times that of the best item in a cell is pruned.

Following (Collins, 1997), words occurring fewer than four times in training were replaced with the symbol *UNKNOWN* and tagged with the output of the part-of-speech tagger described in (Ratnaparkhi, 1996). Tree templates occurring only once in training were ignored entirely.

We first compared the parser with (Hwa, 1998): we trained the model on sentences of length 40 or less in sections 02–09 of the Penn Treebank, down to parts of speech only, and then tested on sentences of length 40 or less in section 23, parsing from part-of-speech tag sequences to fully bracketed parses. The metric used was the percentage of guessed brackets which did not cross any correct brackets. Our parser scored 84.4% compared with 82.4% for (Hwa, 1998), an error reduction of 11%.

Next we compared our parser against lexicalized PCFG parsers, training on sections 02–21 and testing on section 23. The results are shown in Figure 6.

These results place our parser roughly in the middle of the lexicalized PCFG parsers. While the results are not state-of-the-art, they do demonstrate the viability of TAG as a framework for statistical parsing. With

	≤ 40 words					≤ 100 words				
	LR	LP	CB	0 CB	≤ 2 CB	LR	LP	CB	0 CB	≤ 2 CB
(Magerman, 1995)	84.6	84.9	1.26	56.6	81.4	84.0	84.3	1.46	54.0	78.8
(Collins, 1996)	85.8	86.3	1.14	59.9	83.6	85.3	85.7	1.32	57.2	80.8
present model	86.9	86.6	1.09	63.2	84.3	86.2	85.8	1.29	60.4	81.8
(Collins, 1997)	88.1	88.6	0.91	66.5	86.9	87.5	88.1	1.07	63.9	84.6
(Charniak, 2000)	90.1	90.1	0.74	70.1	89.6	89.6	89.5	0.88	67.6	87.7

Figure 6: Parsing results. LR = labeled recall, LP = labeled precision; CB = average crossing brackets, 0 CB = no crossing brackets, ≤ 2 CB = two or fewer crossing brackets. All figures except CB are percentages.

improvements in smoothing and cleaner handling of punctuation and coordination, perhaps these results can be brought more up-to-date.

6 Conclusion: related and future work

(Neumann, 1998) describes an experiment similar to ours, although the grammar he extracts only arrives at a complete parse for 10% of unseen sentences. (Xia, 1999) describes a grammar extraction process similar to ours, and describes some techniques for automatically filtering out invalid elementary trees.

Our work has a great deal in common with independent work by Chen and Vijay-Shanker (2000). They present a more detailed discussion of various grammar extraction processes and the performance of supertagging models (B. Srinivas, 1997) based on the extracted grammars. They do not report parsing results, though their intention is to evaluate how the various grammars affect parsing accuracy and how k-best supertagging afffects parsing speed.

Srinivas's work on supertags (B. Srinivas, 1997) also uses TAG for statistical parsing, but with a rather different strategy: tree templates are thought of as extended parts-of-speech, and these are assigned to words based on local (e.g., n-gram) context.

As for future work, there are still possibilities made available by TAG which remain to be explored. One, also suggested by (Chen and Vijay-Shanker, 2000), is to group elementary trees into families and relate the trees of a family by transformations. For example, one would imagine that the distribution of active verbs and their subjects would be similar to the distribution of passive verbs and their notional subjects, yet they are treated as independent in the current model. If the two configurations could be related, then the sparseness of verb-argument dependencies would be reduced.

Another possibility is the use of multiply-anchored trees. Nothing about PTAG requires that elementary trees have only a single anchor (or any anchor at all), so multiply-anchored trees could be used to make, for example, the attachment of a PP dependent not only on the preposition (as in the current model) but the lexical head of the prepositional object as well, or the attachment of a relative clause dependent on the embedded verb as well as the relative pronoun. The smoothing method described above would have to be modified to account for multiple anchors.

In summary, we have argued that TAG provides a cleaner way of looking at statistical parsing than lexicalized PCFG does, and demonstrated that in practice it performs in the same range. Moreover, the greater flexibility of TAG suggests some potential improvements which would be cumbersome to implement using a lexicalized CFG. Further research will show whether these advantages turn out to be significant in practice.

Acknowledgements

References

B. Srinivas. 1997. *Complexity of lexical descriptions: relevance to partial parsing*. Ph.D. thesis, Univ. of Pennsylvania.

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP 1997)*, pages 194–201.

John Carroll and David Weir. 1997. Encoding frequency information in lexicalized grammars. In *Proceedings of the Fifth International Workshop on Parsing Technologies (IWPT '97)*, pages 8–17.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL2000)*, pages 132–139.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of COLING-ACL '98*, pages 225–231.

John Chen and K. Vijay-Shanker. 2000. Automated extraction of TAGs from the Penn Treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 65–76.

Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191.

Michael Collins. 1997. Three generative lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, Univ. of Pennsylvania.

Joshua Goodman. 1997. Global thresholding and multiple-pass parsing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP-2)*, pages 11–25.

Rebecca Hwa. 1998. An empirical evaluation of probabilistic lexicalized tree insertion grammars. In *Proceedings of COLING-ACL '98*, pages 557–563.

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

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283.

Günter Neumann. 1998. Automatic extraction of stochastic lexicalized tree grammars from treebanks. In *Proceedings of the 4th International Workshop on TAG and Related Formalisms (TAG+4)*, pages 120–123.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 151–158.

Adwait Ratnaparkhi. 1996. A maximum-entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–10.

Philip Resnik. 1992. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-92)*, pages 418–424.

Yves Schabes and Stuart M. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

Yves Schabes and Richard C. Waters. 1995. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21:479–513.

Yves Schabes and Richard Waters. 1996. Stochastic lexicalized tree-insertion grammar. In H. Bunt and M. Tomita, editors, *Recent Advances in Parsing Technology*, pages 281–294. Kluwer Academic Press, London.

Yves Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-92)*, pages 426–432.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, pages 398–403.