6

Statistical parsing with an automatically extracted tree adjoining grammar

David Chiang

6.1 Introduction

Why use tree adjoining grammars (TAG) for statistical parsing? It might be thought that its added formal power makes parameter estimation unnecessarily difficult; or that whatever benefits it provides—the ability to model unbounded cross-serial dependencies, for example—are inconsequential for statistical parsing, which is concerned with the probable rather than the possible.

But just as TAG is not by itself a complete linguistic theory, but a formalism for specifying linguistic theories, it should not be viewed as a statistical model but a formalism for specifying statistical models. The advantage that TAG has over CFG is that it assigns richer *structural descriptions* to sentences; specifically, in addition to parse trees, it assigns *derivation trees* (defined below) on which features of a parsing model can be defined.

In this chapter we explore the use of TAG for statistical parsing. We start by examining PCFG-based parsers which use head-lexicalization to capture bilexical dependencies, beginning with the work of Magerman (1995) and continued by Charniak (1997, 2000) and Collins (1996,

Data-Oriented Parsing.
Rens Bod, Remko Scha, and Khalil Sima'an.

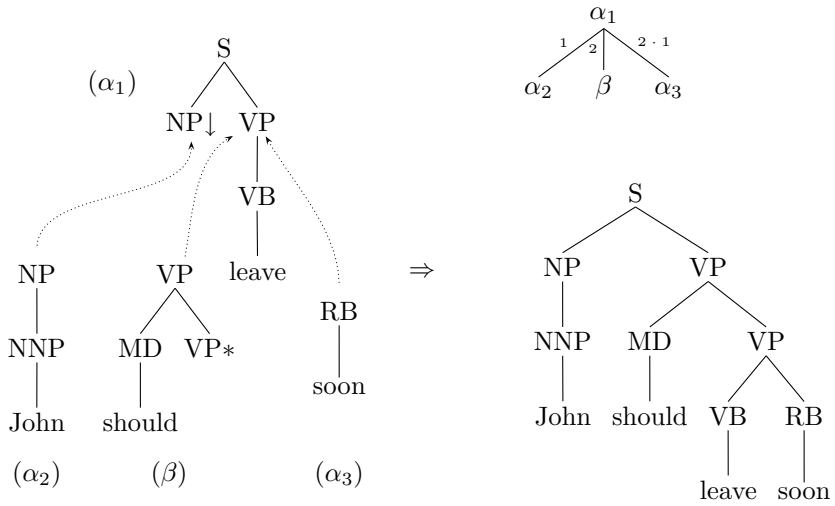

FIGURE 1 Grammar and derivation for "John should leave tomorrow."

1997, 1999), and observe that these models are in fact closely related to probabilistic lexicalized TAG (Schabes 1992, Resnik 1992). We introduce a variant of probabilistic TAG that captures the same bilexical dependencies that these PCFG-based models do (a possibility noted by Resnik (1992) early on), but with less notational overhead, and demonstrate that it can be used to parse with comparable accuracy.

We argue on the way that the use of probabilistic TAG provides two benefits over PCFG: first, it naturally captures some dependencies that must be encoded *ad hoc* into a PCFG, including dependencies which the above PCFG-based parsers do not capture; second, the derivation trees a TAG parser computes in addition to parse trees may be useful for further processing (for example, translation or semantic interpretation). Finally, we examine some recent parsers outside of the lexicalized PCFG family and discuss how TAG might be of use in these systems as well.

6.2 The formalism

The formalism we use is a variant of lexicalized tree insertion grammar (TIG), which is, in turn, a restricted version of lexicalized TAG (Schabes and Waters 1995). In our variant there are three composition operations: *substitution* (α_2 in Figure 1), *adjunction* (β), and *sister-adjunction* (α_3). Auxiliary trees and adjunction are restricted as in TIG: essentially, auxiliary trees with nonempty terminal nodes on both sides of the foot node are not allowed, nor any adjunction operation

that might create such a tree.

Sister-adjunction is not an operation found in standard definitions of TAG, but is similar to the sister-adjunction operation of d-tree substitution grammar (Rambow et al. 1995) and the modification operation of Sima'an's tree-gram model (2000, 2002b). In this operation the root of an initial tree is inserted as a new daughter of an interior node. It is similar to modifier adjunction (Schabes and Shieber 1994) in that multiple trees can be sister-adjoined at a single position. (By contrast, only one auxiliary tree may be adjoined at a single node.) This operation does not add any formal power; we simply introduce it so we can derive the flat structures found in the Penn Treebank.

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. Figure 1 shows an example grammar and the derivation of the sentence "John should leave soon." 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 adjunction site. An arc corresponding to the sister-adjunction of a tree between the ith and i+1st 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*.

Probabilistic TAG (Resnik 1992, Schabes 1992) is a history-based model like PCFG; that is, it decomposes the probability of a tree into probabilities of decisions made in building the tree. In PCFG the decisions are rewritings of nonterminal symbols; in probabilistic TAG the decisions are composition operations. The parameters of a probabilistic TAG are:

$$\begin{aligned} \sum_{\alpha} P_i(\alpha) &= 1 \\ \sum_{\alpha} P_s(\alpha \mid \eta) &= 1 \\ \sum_{\beta} P_a(\beta \mid \eta) + P_a(\mathit{NONE} \mid \eta) &= 1 \end{aligned}$$

where α ranges over initial trees, β over auxiliary 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, $P_a(\mathit{NONE} \mid \eta)$ is the proba-

[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 address i has address $i \cdot j$.

bility of nothing adjoining at η. Joshi and Sarkar (2002) give a fuller introduction to PTAG; Carroll and Weir (1997, 2002) suggest other parameterizations worth exploring as well.

Our variant adds another set of parameters for sister-adjunction:

$$\sum_{\alpha} P_{sa}(\alpha \mid \eta, i, X) + P_{sa}(STOP \mid \eta, i, X) \;=\; 1$$

where α ranges over initial trees, and (η, i) ranges over possible sister-adjunction sites. $P_{sa}(\alpha)$ is the probability of sister-adjoining α, and $P_{sa}(STOP)$ is the probability of no further sister-adjunction. X is the root label of the previous tree (i. e., closer to the head) to sister-adjoin at the site (η, i), or $START$ if none; thus the roots of the trees sister-adjoining at a given site are generated by a first-order Markov process. This is similar to the base-NP model in Collins' model (1997) and the model from BBN's SIFT system (Miller et al. 1998).

The probability of a derivation can then be expressed as a product of the probabilities of the individual operations of the derivation. Thus the probability of the example derivation of Figure 1 would be

$$\begin{aligned} &P_i(\alpha_1) \cdot P_a(NONE \mid \alpha_1(\epsilon)) \cdot P_s(\alpha_2 \mid \alpha_1(1)) \cdot P_a(\beta \mid \alpha_1(2)) \cdot \\ &\quad P_{sa}(\alpha_3 \mid \alpha_1(2), 1, START) \cdot P_{sa}(STOP \mid \alpha_2(2), 1, \mathrm{RB}) \cdot \\ &\quad P_{sa}(STOP \mid \alpha_2(\epsilon), 0, START) \cdot \ldots \end{aligned}$$

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

6.3 Inducing a stochastic grammar from the Treebank

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. For there are many TAG derivations which can yield the same derived tree, even with respect to a single grammar. We need, then, to reconstruct TAG derivations somehow from the training data.

One approach, taken by Hwa (1998), is to choose some grammar general enough to parse the whole corpus and obtain a maximum-likelihood estimate using Expectation-Maximization (EM). A different approach, taken by Magerman (1995) and others for lexicalized PCFGs and Neumann (1998) and others (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 and then EM to reestimate its parameters.)

In fact, we can follow Magerman's approach very closely. In a lexicalized TAG, because each composition brings together two lexical items, every composition probability involves a bilexical dependency. Given a CFG and a Magerman-style head-percolation scheme, an equivalent TAG can be constructed whose derivations mirror the dependency analysis implicit in the head-percolation scheme, as shown below.

6.3.1 Reconstructing derivations

For each node η, the head-percolation and argument/adjunct rules classify exactly one child of η as a head and the rest as either arguments or adjuncts. We use Collins' rules with few modifications, but we treat coordination specially: if an X dominates a CC, and the rightmost CC has an X to its left and to its right, then that CC is marked as the head and the two nearest Xs on either side as arguments.

Using this classification into heads, arguments, and adjuncts, 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 sister-adjoined initial tree.
2. If η is an argument, excise the subtree rooted at η to form an initial tree, leaving behind a substitution node.
3. But if η is an argument and η' is the nearest ancestor with the same label, and η is the rightmost descendant of η', and all the intervening nodes, including η', are heads, excise the segment from η' down to η to form an auxiliary tree, leaving behind a head node.

Rules (1) and (2) produce the desired result; rule (3) changes things somewhat by making trees with recursive arguments into auxiliary trees. For example, if applied to the derived tree in Figure 1, it would be responsible for extracting β. In the present implementation, in fact, it is restricted to nodes labeled VP. The complicated restrictions on η' are simply to ensure that a well-formed TIG derivation is produced.

6.3.2 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 6.2. But since the number of possible composition operations 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 stripped of its lexical anchor), then the anchor. The probabilities are decomposed as follows:

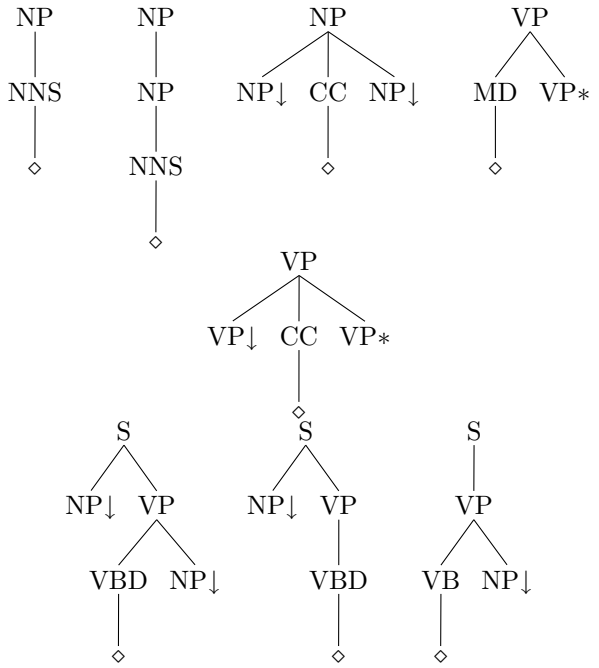

FIGURE 2 A few of the more frequently-occurring tree templates. ◇ marks where the lexical anchor is inserted.

$$\begin{aligned} P_i(\alpha) &= P_{i_1}(\tau_\alpha) P_2(w_\alpha \mid \tau_\alpha) \\ P_s(\alpha \mid \eta) &= P_{s_1}(\tau_\alpha \mid \eta) \cdot P_2(w_\alpha \mid \tau_\alpha, t_\eta, w_\eta) \\ P_a(\beta \mid \eta) &= P_{a_1}(\tau_\beta \mid \eta) \cdot P_2(w_\beta \mid \tau_\beta, t_\eta, w_\eta) \\ P_{sa}(\alpha \mid \eta, i, X) &= P_{sa_1}(\tau_\alpha \mid \eta, i, X) \cdot P_2(w_\alpha \mid \tau_\alpha, t_\eta, w_\eta, X) \end{aligned}$$

where τ_α is the tree template of α and w_α is the lexical anchor of α, and similarly for β; w_η is the lexical anchor of the elementary tree containing η, and t_η is the part-of-speech (POS) tag of that anchor. Note that the same probability P_2 is used for all three composition operations: when X does not have a meaningful value, the value $START$ is used.

These probabilities each have three backoff levels:

	$P_{s_1,a_1}(\gamma \mid \cdots)$	$P_{sa_1}(\gamma \mid \cdots)$	$P_2(w \mid \cdots)$
1	$\tau_\eta, w_\eta, \eta_\eta$	$\tau_\eta, w_\eta, \eta_\eta, i, X$	$\tau_\gamma, t_\eta, w_\eta, X$
2	τ_η, η_η	$\tau_\eta, \eta_\eta, i, X$	τ_γ, t_η, X
3	$\bar{\tau}_\eta, \eta_\eta$	$\bar{\tau}_\eta, \eta_\eta, i$	τ_γ
4	$\emptyset$	$\emptyset$	t_γ

where τ_η is the tree template of the elementary tree containing η, $\bar{\tau}_\eta$ is τ_η stripped of its anchor's POS tag, and η_η is the address of η in its elementary tree; τ_γ is the tree template of γ, and t_γ is the POS tag of its anchor. The backed-off models are combined by linear interpolation:

$$e = \lambda_1 e_1 + (1 - \lambda_1)(\lambda_2 e_2 + (1 - \lambda_2)(\lambda_3 e_3 + (1 - \lambda_3) e_4))$$

where e_i is the estimate at level i, and the λ_i are computed by a combination of formulas used by Collins (1999) and Bikel et al. (1997):

$$\lambda_i = \left(1 - \frac{d_{i-1}}{d_i}\right)\left(\frac{1}{1 + 5u_i/d_i}\right)$$

where d_i is the number of occurrences in training of the context at level i ($d_0 = 0$), and u_i is the number of unique outcomes for that context seen in training.

To handle unknown words, we treat all words seen five or fewer times in training as a single symbol `*UNKNOWN*`, following Collins (1997).

6.4 Experimental details

6.4.1 Extracting the grammar

When we run the algorithm given in Section 6.3.1 on sections 02–21 of the Penn Treebank, the resulting grammar has 50,628 lexicalized trees (with words seen five or fewer times replaced with `*UNKNOWN*`). However, if we consider elementary tree templates, the grammar is quite manageable: 2104 tree templates, of which 1261 occur more than once

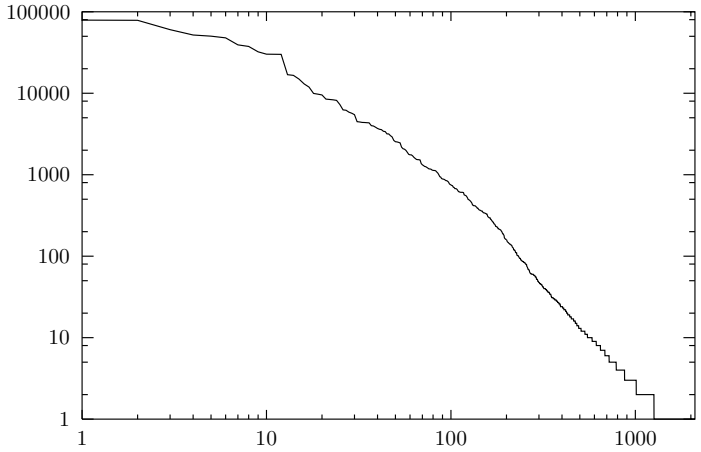

FIGURE 3 Distribution of tree templates: frequency versus rank (log-log)

(see Figure 3). A few of the most frequent tree templates are shown in Figure 2.

So the extracted grammar is fairly compact, but how complete is it? Ideally the size of the grammar would converge, but if we plot its growth during training (Figure 4), we see that even after training on 1 million words, new elementary tree templates continue to appear at the rate of about four every 1000 words, or one every ten sentences.

We do not consider this effect to be seriously detrimental to parsing. Since 90% of unseen sentences can be parsed perfectly with the extracted grammar, its coverage is good enough potentially to parse new data with state-of-the-art accuracy. Note that for the remaining 10% it is still quite possible for the grammar to derive a perfect parse, since there can be many TAG derivations which yield the same derived tree. Nevertheless, we would like to know the source of this effect and minimize 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 errors

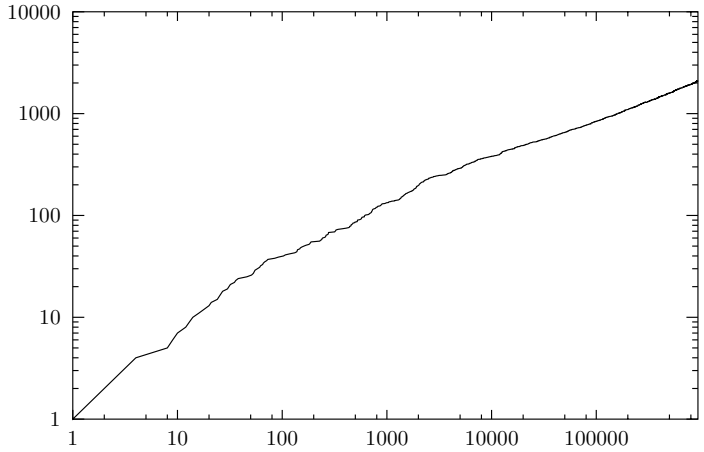

FIGURE 4 Growth of grammar during training: types versus tokens (log-log)

in the text itself. Only twelve appeared to be genuine.[2]

Therefore the extracted grammar is more complete than Figure 4 suggests at first glance. Evidently, however, our extraction heuristics 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.

6.4.2 Parsing with the grammar

We used a CKY-style parser similar to the one described by Schabes and Waters (1996), with a modification to ensure completeness (because foot nodes are treated as empty, which standard CKY does not handle). We also extended the parser to simulate sister-adjunction as regular adjunction.

The parser uses a beam search, assigning a score to each item $[\eta, i, j]$ and pruning away any item with score less than 10^{-5} times that of the best item for that span. The score of an item is its inside probability multiplied by the prior probability $P(\eta)$, following Goodman (1997). $P(\eta)$, in turn, is decomposed as $P(\bar{\tau}_\eta \mid t_\eta, w_\eta) \cdot P(t_\eta, w_\eta)$, so that the first term can be smoothed by linear interpolation (see Section 6.3.2) with the backed-off estimate $P(\bar{\tau}_\eta \mid t_\eta)$, following Collins (1999).

As mentioned above (Section 6.3.2), words occurring five or fewer times in training were replaced with the symbol `*UNKNOWN*`. When any

[2]This survey was performed on an earlier version of the extraction heuristics.

such word w occurs in the test data, it is also replaced with `*UNKNOWN*`. Following Collins (1997), the parser only allows w to anchor templates that have POS tags observed in training with w itself, or templates that have the POS tag assigned to w by MXPOST (Ratnaparkhi 1996); all other templates are thrown out for w. In addition, tree templates occurring only once in training (with any anchor) are thrown out of the grammar entirely.

6.5 Results and discussion

6.5.1 Comparison with previous work in TAG

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 TAG extraction system similar to ours, and Sarkar (2001) reports results of parsing using its output grammar, but since his work focuses on combining annotated and unannotated training data, these results are not directly comparable to ours. Finally, Chen and Vijay-Shanker (2000) describe another similar TAG extraction system, but do not report parsing results.

We can, however, directly compare our parser to Hwa's (1998). To do this, we trained the model on sentences of length 40 or less in sections 02–09 of the Penn Treebank, down to POS tags only, and then tested on sentences of length 40 or less in section 23, parsing from POS tag sequences to fully bracketed parses. The metric used was the percentage of guessed brackets which did not cross any correct brackets:

Hwa (1998)	82.4
present model	84.7

The present model obtains an error reduction of 13% compared with Hwa's. (It should be noted that Hwa's model does not make use of the nonterminal labels in the training data, so in a sense our model was trained on more data.)

6.5.2 Comparison with lexicalized PCFG parsers

Next we compared our parser against lexicalized PCFG parsers, training on sections 02–21 and testing on section 23. The results (Figure 5) show that our parser lies roughly midway between the earliest (Magerman 1995) and latest (Charniak 2000) of this class.[3] While these results are not state-of-the-art, they do verify the hypothesis that a probabilistic TAG parser should perform at the same level as a lexicalized PCFG parser.

[3]Note that these figures are an improvement over those of an earlier version (Chiang 2000).

	LR	LP	CB	0 CB	≤ 2 CB
≤ 40 words					
Magerman (1995)	84.6	84.9	1.26	56.6	81.4
Charniak (1997)	87.5	87.4	1.0	62.1	86.1
present model	87.7	87.7	1.03	65.1	84.8
Collins (1999)	88.5	88.7	0.92	66.7	87.1
Charniak (2000)	90.1	90.1	0.74	70.1	89.6
≤ 100 words					
	LR	LP	CB	0 CB	≤ 2 CB
Magerman (1995)	84.0	84.3	1.46	54.0	78.8
Charniak (1997)	86.7	86.6	1.20	59.5	83.2
present model	87.0	86.9	1.22	62.0	82.2
Collins (1999)	88.1	88.3	1.06	64.0	85.1
Charniak (2000)	89.6	89.5	0.88	67.6	87.7

FIGURE 5 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.

Furthermore, there are some dependency analyses which are encodable by TAGs but not 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 6a). Nearly a quarter of nonempty subjects appear in such a configuration. 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 the cases listed above requires special treatment in a PTAG model, because each composition probability involves not only a bilexical dependency but a "biarboreal" (tree-tree) dependency. That

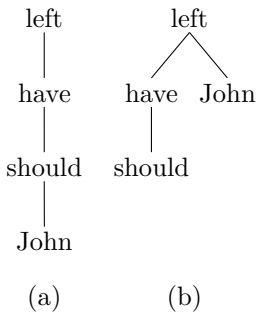

FIGURE 6 Bilexical dependencies for "John should have left."

is, PTAG generates an entire elementary tree at once, conditioned on the entire elementary tree being modified. Thus some 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, this additional context-sensitivity brings with it increased data sparseness; we suspect that our model does not do better than lexicalized PCFG models because it is not using this extra information very robustly. Fine-tuning the backoff model might bring accuracy closer to the state-of-the-art, but it may be more productive to look beyond history-based models to models which provide more flexibility.

6.5.3 Beyond history-based models

Our parser does not fare as well when compared with recent parsers which, unlike our parser and lexicalized PCFG–based parsers, are not history-based models:

	≤ 40 words		≤ 100 words	
	LR	LP	LR	LP
present model	87.7	87.7	87.0	86.9
Collins (2000)	90.1	90.4	89.6	89.9
Bod (2001)	90.6	90.8	89.6	89.5

The Data-Oriented Parsing (DOP1) model (Bod and Scha 2002) is related to ours in that both are based on probabilistic tree grammars; Hoogweg's TIGDOP model (2000, 2002) is even more closely related, being based on TIG. But the key difference is that our parser computes the best single derivation for the input sentence, whereas DOP1 computes the best derived tree for the input sentence, by summing over many derivations.

We see two main benefits of our method. First, computing the highest-probability single derivation is much more efficient than computing the highest-probability derived tree. The latter is, in fact, NP-hard (Sima'an 1999, Sima'an 2002a). Bod uses random sampling to obtain a 5 million elementary tree subset of the much larger full grammar, and then computes only the 1,000 best derivations, summing those which yield the same derived tree. Even with the search space thus reduced, this procedure is still very computationally expensive.

Second, even though a parser which finds the best single derivation does less work, it delivers more, because it computes a derivation tree in addition to a parse tree. This is important for applications in which the derivation is used for later processing. For example, in synchronous TAG (Shieber 1994) it is used as an interlingua of sorts for machine translation. In the LTAG semantics of Kallmeyer and Joshi (1999), interpretations are computed compositionally, not on derived trees, but on derivation trees. Generally speaking, under any interpretation of TAG which assigns some linguistic significance to derivation trees, a parser which computes only derived trees is inadequate.

The primary advantage of DOP over our model is that DOP makes use of multi-anchored elementary trees. It would be possible to do this in our model as well, simply by modifying the extraction heuristics to produce multi-anchored elementary trees. For example, we might put a preposition and the headword of its object into the same elementary tree, so that the attachment of a PP would be sensitive to its prepositional object.

Our initial experiments in this direction, however, have yielded a decrease in accuracy. The problem seems to be that our smoothing method does not handle the additional data sparseness well. One might think about backing off from generating multi-anchored trees to generating single-anchored trees; but since this means backing off from one kind of derivation to another kind of derivation, this makes no sense if we are looking for the best single derivation.

On the other hand, since DOP sums over all possible derivations of a parse tree, it can combine probability estimates of parses which are generated in chunks of varying size. Thus summing over derivations appears to accomplish what our smoothing method does not. (Note that the model described by Sima'an (2000, 2002b) employs neither smoothing nor summing over derivations, and performs worse than models which do either: 83/85% labeled recall/precision for sentences of length ≤ 40.)

Another system which makes effective use of larger amounts of context is described by Collins (2000). This system takes the parses pro-

duced by a PCFG-based parser (Collins 1999) and uses a second model to rerank them. This second model treats trees not as sequences of decisions, but as arbitrary sets of features (not in the sense of annotations on nodes, but characteristics of a tree which the model looks at to compute a score). It is thus able to take into account more or less context as appropriate.

There is no reason why such a method could not be applied to reranking TAG derivations instead of CFG derivations. In such a system one could define features which span more than two elementary trees (for example, relating a preposition, its object, and the NP or VP it modifies), without the complication of multi-anchored trees or other alterations to the grammar. The advantage of TAG over CFG in this case would be that TAG derivations provide a structure on which more complex features can be more easily defined (for example, relating subjects and objects, which do not always appear in the same relative configuration on the derived tree). And unlike DOP, such a system would still compute derivations, retaining the advantages mentioned above for applications.

6.5.4 An experiment on the Penn Chinese Treebank

To see how this system would adapt to a different corpus in a different language, we replaced the head rules and argument/adjunct rules with rules appropriate for the Penn Chinese Treebank (Xia et al. 2000). These were adapted from rules constructed by Xia (1999).

We also made the following changes to the experimental setup:

- We lowered the unknown word threshold from five to one because the new training set was smaller.
- The POS tagger for unknown words had to be retrained on the new corpus.
- A beam width of 10^{-4} was used instead of 10^{-5}.

We then trained the parser on sections 001–270 of the Penn Chinese Treebank (84,873 words) and tested it on sections 271–300 (6776 words). To provide a basis for comparison with performance on English, we also trained the parser on sections 02–03 of the WSJ corpus (82,592 words) and tested it on the first 400 sentences of section 23 (9473 words). Note that because of the relatively small datasets used, cross-validation would be desirable for future studies.

The results (Figure 7) show that although there is a sizable gap between performance on English and Chinese,[4] it is quite usable on a

[4]This gap is wider than it was for an earlier version (Bikel and Chiang 2000), even though accuracy for both languages improved.

				≤ 40 words		
Model	Corpus	LR	LP	CB	0 CB	≤ 2 CB
present	WSJ-small	84.0	83.6	1.34	50.4	79.4
present	**Xinhua**	77.8	78.1	1.98	49.2	69.8
				≤ 100 words		
Model	Corpus	LR	LP	CB	0 CB	≤ 2 CB
present	WSJ-small	83.4	83.1	1.53	48.5	76.8
present	**Xinhua**	75.3	75.7	2.82	43.4	62.7

FIGURE 7 Parsing results. Abbreviations are as in Figure 5. Xinhua: trained on Penn Chinese Treebank sections 001–270, tested on sections 271–300. WSJ-small: trained on Penn Treebank, Wall Street Journal sections 02–03, tested on section 23, sentences 1–400.

language other than the one it was developed on. Indeed, this parser is currently being used to augment the Penn Chinese Treebank, providing rough parses which human annotators can correct up to twice as fast as annotating from scratch (Chiou et al. 2001).

6.6 Conclusion

We have shown that probabilistic TAG can be used for statistical parsing with accuracy comparable to that of models based on lexicalized PCFG, and that rule-based reconstruction of TAG derivations from bracketed training data gives better accuracy than an EM-based method (Hwa 1998).

Moreover, we have argued that TAG has advantages over CFG in two areas: first, TAG derivations provide a convenient structure on which features (like bilexical dependencies) can be easily defined; second, TAG derivations provide extra information which is useful for applications such as translation. Since neither of these advantages is bound to a particular statistical model, we hope to continue this work by exploring the usefulness of TAG under other approaches to the parsing problem and its applications.

Acknowledgements

This research is supported in part by ARO grant DAAG55971-0228 and NSF grant SBR-89-20230-15. Thanks to Mike Collins, Anoop Sarkar, Dan Bikel, Fei Xia, Aravind Joshi, and the anonymous reviewers for their valuable help. *S. D. G.*

References

Bikel, Daniel M., and David Chiang. 2000. Two statistical parsing models applied to the Chinese Treebank. In *Proceedings of the Second Chinese Language Processing Workshop*, 1–6. Hong Kong.

Bikel, Daniel M., 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)*, 194–201. Washington, DC.

Bod, Rens. 2001. What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings of ACL-2001*, 66–73. Toulouse.

Bod, Rens, and Remko Scha. 2002. A DOP model for phrase-structure trees. In *Data-Oriented Parsing*, ed. Rens Bod, Remko Scha, and Khalil Sima'an. Stanford, CA: CSLI Publications.

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

Carroll, John, and David Weir. 2002. Encoding frequency information in stochastic parsing models. In *Data-Oriented Parsing*, ed. Rens Bod, Remko Scha, and Khalil Sima'an. Stanford, CA: CSLI Publications.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 598–603. AAAI Press/MIT Press.

Charniak, Eugene. 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)*, 132–139. Seattle.

Chen, John, 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)*, 65–76. Trento.

Chiang, David. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Assocation for Computational Linguistics*, 456–463. Hong Kong.

Chiou, Fu-Dong, David Chiang, and Martha Palmer. 2001. Facilitating treebank annotation using a statistical parser. In *Proceedings of HLT 2001*. San Diego, CA. Poster session.

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

Collins, Michael. 1997. Three generative lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Assocation for Computational Linguistics (ACL-EACL '97)*, 16–23. Madrid.

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Doctoral dissertation, Univ. of Pennsylvania.

Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 175–182. Stanford, CA. Morgan Kaufmann.

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

Hoogweg, Lars. 2000. Extending DOP1 with the Insertion Operation. Master's thesis, University of Amsterdam.

Hoogweg, Lars. 2002. Extending DOP with adjunction. In *Data-Oriented Parsing*, ed. Rens Bod, Remko Scha, and Khalil Sima'an. Stanford, CA: CSLI Publications.

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

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

Joshi, Aravind, and Anoop Sarkar. 2002. Tree-Adjoining Grammars and its application to statistical parsing. In *Data-Oriented Parsing*, ed. Rens Bod, Remko Scha, and Khalil Sima'an. Stanford, CA: CSLI Publications.

Kallmeyer, Laura, and Aravind K. Joshi. 1999. Factoring predicate argument and scope semantics: underspecified semantics with LTAG. In *Proceedings of the 12th Amsterdam Colloquium*, ed. Paul Dekker, 169–174.

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

Miller, Scott, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 1998. SIFT – Statistically-derived Information From Text. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Washington, DC.

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

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

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

Resnik, Philip. 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)*, 418–424. Nantes.

Sarkar, Anoop. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, 175–182. Pittsburgh, PA.

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

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

Schabes, Yves, 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.

Schabes, Yves, and Richard C. Waters. 1996. Stochastic lexicalized tree-insertion grammar. In *Recent Advances in Parsing Technology*,

ed. H. Bunt and M. Tomita. 281–294. London: Kluwer Academic Press.

Shieber, Stuart M. 1994. Restricting the weak generative capacity of synchronous tree-adjoining grammars. *Computational Intelligence* 10(4):371–385. Special Issue on Tree Adjoining Grammars.

Sima'an, K. 2000. Tree-gram parsing: lexical dependencies and structural relations. In *Proceedings of the 38th Annual Meeting of the Assocation for Computational Linguistics*, 53–60. Hong Kong.

Sima'an, Khalil. 1999. *Learning Efficient Disambiguation*. Doctoral dissertation, University of Amsterdam.

Sima'an, Khalil. 2002a. Computational complexity of disambiguation under DOP. In *Data-Oriented Parsing*, ed. Rens Bod, Remko Scha, and Khalil Sima'an. Stanford, CA: CSLI Publications.

Sima'an, Khalil. 2002b. A head-driven data-oriented approach to lexical dependency. In *Data-Oriented Parsing*, ed. Rens Bod, Remko Scha, and Khalil Sima'an. Stanford, CA: CSLI Publications.

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

Xia, Fei, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Fu-Dong Chiou, Shizhe Huang, Tony Kroch, and Mitch Marcus. 2000. Developing guidelines and ensuring consistency for Chinese text annotation. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC-2000)*. Athens, Greece.