Chapter 4

AUTOMATED EXTRACTION OF TAGS FROM THE PENN TREEBANK

John Chen
Department of Computer Science, Columbia University
11214 Amsterdam Avenue, New York, NY 10027-7003
jchen@cs.columbia.edu

Vijay K. Shanker
Department of Computer and Information Sciences, University of Delaware
103 Smith Hall, Newark, DE 19716
vijay@cis.udel.edu

Abstract The accuracy of statistical parsing models can be improved with the use of lexical information. Statistical parsing using Lexicalized tree adjoining grammar (LTAG), a kind of lexicalized grammar, has remained relatively unexplored. We believe that is largely in part due to the absence of large corpora accurately bracketed in terms of a perspicuous yet broad coverage LTAG. Our work attempts to alleviate this difficulty. We extract different LTAGs from the Penn Treebank. We show that certain strategies yield an improved extracted LTAG in terms of compactness, broad coverage, and supertagging accuracy. Furthermore, we perform a preliminary investigation in smoothing these grammars by means of an external linguistic resource, namely, the tree families of the XTAG grammar, a hand built grammar of English.

1. Introduction

Not only have lexicalized grammars been shown to be linguistically appealing, but they have also been shown to be desirable for parsing disambiguation as well. Among others, for example, Charniak (1996) and Collins (1996) have found that lexicalizing a probabilistic model substantially increases parsing accuracy. As introduced in Schabes et al. (1988), lexicalized tree adjoining grammar (LTAG) is a lexicalized grammar formalism in which lexical items are associated with sets of grammatical structures. Resnik (1992) shows that parsing disambiguation can be aided by statistical knowledge of cooccurrence relationships between LTAG structures. Srinivas (1997) and Chen et al. (1999) show that considerable parsing disambiguation is accomplished by assigning LTAG structures to words in the sentence using part of speech tagging techniques (supertagging).

An LTAG grammar G and a means of estimating parameters associated with G are prerequisites for probabilistic LTAG. Schabes (1992) shows how this may be done through grammar induction from an unbracketed corpus. The number of parameters that must be estimated, however, is prohibitively large for all but the most simple grammars. In contrast, the XTAG group has developed XTAG, a complex, relatively broad coverage grammar for English (XTAG-Group, 2001) . It is difficult, however, to estimate parameters with XTAG because it has been verified to accurately parse only relatively small corpora, such as the ATIS corpus. Marcus et al. (1993) describes the Penn Treebank, a corpus of parsed sentences that is large enough to estimate statistical parameters. From the treebank, Srinivas (1997) heuristically derives a corpus of sentences where each word is annotated with an XTAG tree, thus allowing statistical estimation of an LTAG. This method entails certain drawbacks: the heuristics make several mistakes, some unavoidable because of discrepancies between how XTAG and the Penn Treebank annotate the same grammatical constructions, or because XTAG does not cover all of the grammatical phenomena found in the Penn Treebank. Furthermore, these corpus mistakes usually propagate to the statistical model.

In this work, we explore extraction of an LTAG from the Penn Treebank. This allows us not only to obtain a wide coverage LTAG but also one for which statistical parameters can be reliably estimated. First, we develop various methods for extracting an LTAG from the treebank with the aim of being consistent with current principles for developing LTAG grammars such as XTAG. Second, we evaluate each grammar resulting from these methods in terms of its size, its coverage on unseen data, and its supertagging performance. Third, we introduce a preliminary method to extend an extracted grammar in order to improve coverage. Fourth, we situate our current work with respect to other approaches to tree extraction. Lastly, we present our conclusions and suggestions for future work.

2. Tree Extraction Procedure

In this section, we first describe the goals behind a tree extraction procedure and then describe the tree extraction procedure and its variations.

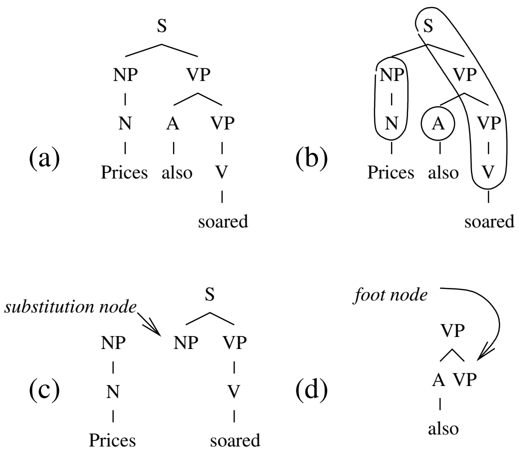

Figure 4.1. (a) Sentential structure. (b) Sentential structure where nonterminals belonging to the same trunk have been circled. (c) Localizing argument dependencies in the same elementary tree. (d) Each instance of recursion is factored into a separate elementary tree.

An LTAG G is defined as a set of *elementary trees* T which are partitioned into a set I of *initial trees* and a set A of *auxiliary trees*. The frontier of each elementary tree is composed of a lexical *anchor*; the other nodes on the frontier are *substitution nodes*, and, in the case of an auxiliary tree, one node on the frontier will be a *foot node*. The foot node of a tree β is labeled identically with the root node of β. The *spine* of an auxiliary tree is the path from its root to its foot node. This is distinct from the *trunk* of an elementary tree which is the path from its root node to the lexical anchor.

Although the formalism of LTAG allows wide latitude in how trees in T may be defined, several linguistic principles generally guide their formation. First, dependencies, including long distance dependencies, are typically localized in the same elementary tree by appropriate grouping of syntactically or semantically related elements; i.e., complements of a lexical item are included in the same tree as shown in Figure 4.1(c). Second, recursion is factored into separate auxiliary trees as shown in Figure 4.1(d).

The genesis of a tree γ lexicalized by a word $w \in S$, where S is a bracketed sentence in the Penn Treebank, using our tree extraction procedure proceeds as follows. First, a *head percolation table* is used to determine the trunk of γ. Introduced by Magerman (1995), a head percolation table assigns to each node in S a *headword* using local structural information. The trunk of γ is defined to be that path through S whose nodes are labeled with the headword w, examples of which are shown in Figure 4.1(b). Each node η' that is immediately dominated by a node η on the trunk may either be itself on the trunk, a complement of the trunk's headword – in which case it belongs to γ, or an adjunct of the trunk's headword – in which case it belongs to another (auxiliary) tree β which modifies γ.

It is therefore necessary to determine a node's status as a complement or adjunct. Collins (1997) introduces a procedure which determines just this from the treebank according to the node's label, its semantic tags, and local structural information. As described in Marcus et al. (1994), a node's semantic tags provide useful information in determining the node's status, such as grammatical function and semantic role. Our procedure for identifying complements or adjuncts closely follows Collins's method. The main differences lie in our attempt to treat those nodes as complements which are typically localized in LTAG trees. A critical departure from Collins is in the treatment of landing site of wh-movement. Collins's procedure treats the NP landing site as the head and its sibling (typically labelled S) as a complement. In our procedure for extracting LTAG trees, we project from a lexical item up a path of heads. Then, by adopting Collins's treatment, the landing site would be on the path of projection and from our extraction procedure, the wh-movement would not be localized. Hence, we treat the sibling (S node) of the landing site as the head child and the NP landing site as a complement. Figure 4.4(c) shows an example of a lexicalized tree we extract that localizes long-distance movement.

We have conducted experiments on two procedures for determining a node's status as complement or adjunct. The first procedure that we consider, "CA1," uses the label and semantic tags of node η and η's parent in a two step procedure. In the first step, exactly this information is used as an index into a manually constructed table, a *complement adjunct table*, which determines complement or adjunct status. "IF current node is PP-DIR AND parent node is VP THEN assign adjunct to current node" is an example of an entry in this table. The table is sparse; should the index not be found in the table then the second step of the procedure is invoked:

1 Nonterminal PRN is an adjunct.

2 Nonterminals with semantic tags NOM, DTV, LGS, PRD, PUT, SBJ are complements.

3 Nonterminals with semantic tags ADV, VOC, LOC, PRP are adjuncts.

4 If none of the other conditions apply, the nonterminal is an adjunct.

Whereas CA1 uses the label and semantic tags of a node η and its parent η', the procedure described in Xia (1999), "CA2," uses the label and semantic tags of a node η, its *head sibling* η_h, and *distance* between η and η_h in order to determine the complement or adjunct status of node η. CA2 relies on two manually constructed tables: an argument table and a tagset table. The argument table votes for η being a complement given the label and semantic tags of η and η_h and the distance between them. For example, if η is marked as NP,

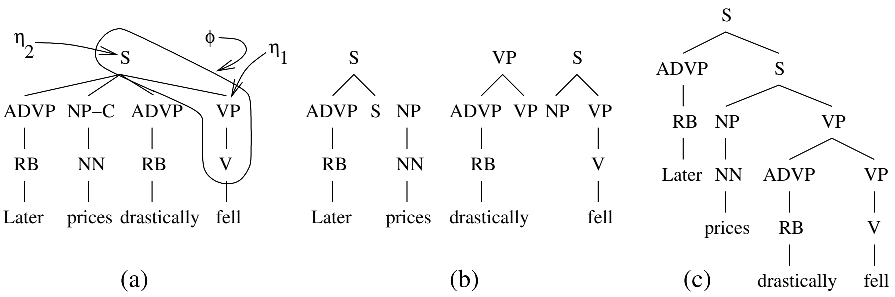

Figure 4.2. (a) Original Treebank bracketing with head sibling η_1 and its parent η_2 both on trunk of headword "fell," and siblings of η_1 marked "-C" for complement or no annotation for adjunct. (b) Extracted trees. (c) Bracketing as defined from extracted trees.

then the argument table votes for η if η_h is labeled VB and if there is a less than four node separation between η and η_h. The tagset table votes for η being a complement based on the semantic tags of η alone. If both the argument table and the tagset table vote that η should be a complement, it is labeled as such. Otherwise, it is labeled as an adjunct.

A recursive procedure is used to extract trees bottom up given a particular treebank bracketing. Figure 4.2(a) shows one step in this process. Among all of the children of node η_2, one child η_1 is selected using the head percolation table so that the trunk ϕ associated with η_1 is extended to η_2. η_1's siblings are subsequently marked as either complement or adjunct. Complement nodes are attached to trunk ϕ and the trees that they dominate become initial trees. Adjuncts are factored into auxiliary trees such that those farthest from η_1 adjoin to η_2 and those closest to η_1 adjoin to η_1, as seen in Figure 4.2(b). These are *modifier* auxiliary trees, not *predicative* auxiliary trees, which will be discussed later. Although the structure that is defined by the resulting grammar may differ from the original bracketing (see Figure 4.2(c)), none of the modified bracketings contradicts those in the original Treebank structure, unlike the heuristically derived corpus used by Srinivas (1999). This is important for our goal of ultimately comparing a parser based on this grammar to others' parsers trained on the Penn Treebank. This factoring tends to reduce the number of trees in the resulting grammar. For example, the extracted trees in Figure 4.2(b) can be used to represent not only "Later prices drastically fell" but other sentences such as "Prices fell" and "Later prices fell."

Our tree extraction procedure also factors the recursion that is found in conjunction. Conjunction in the Treebank is represented as a flat structure such as Figure 4.3(a). We define an *instance of conjunction* to be a sequence of siblings in the tree $\langle\langle X\rangle_1\ \langle,\rangle_2\ \langle X\rangle_2\ \ldots\ \langle CC\rangle_k\ \langle X\rangle_k\rangle$ where $\langle,\rangle_i$, $\langle X\rangle_i$, and $\langle CC\rangle_i$ are labels of the siblings, and there are k conjuncts. This follows from a

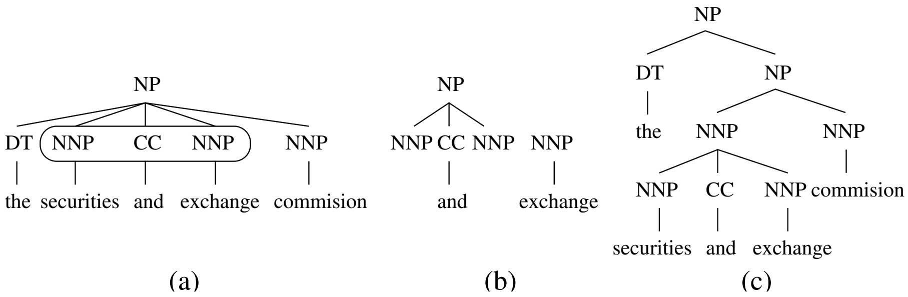

Figure 4.3. (a) Treebank bracketing of conjunction where instance of conjunction is circled. (b) Trees extracted from conjunct. (c) Bracketing as defined from extracted trees.

basic linguistic notion which states that only like categories can be conjoined. When this configuration occurs in a Treebank bracketing, each pair $\langle\langle , \rangle_i \, \langle X \rangle_i\rangle$ (or $\langle\langle CC \rangle_k \, \langle X \rangle_k\rangle$) is factored into elementary trees as follows. The $\langle , \rangle_i$th (or $\langle CC \rangle_k$th) sibling anchors an auxiliary tree β representing the ith (kth) conjunct. The $\langle X \rangle_i$th (or $\langle X \rangle_k$th) sibling anchors an elementary tree that substitutes into β. See Figure 4.3(b) for an example where $k = 1$. After factoring of recursion found in the conjunction and in the adjuncts of Figure 4.3(a), the tree extraction procedure returns a grammar that licenses the structure in Figure 4.3(c).

The technique described above only considers the conjunction of like categories. Although most conjunction is of this nature, it sometimes occurs that constituents with unlike categories are conjoined. In the Penn Treebank, these are annotated with the nonterminal label UCP. Although our current tree extraction procedure does not treat these cases specially as conjunction, a similar strategy may be employed that does so, and in any case they are quite rare.

The commas that were found in instances of conjunction were only one example of numerous cases of punctuation that are found in the treebank. Punctuation symbols are treated as adjuncts. On the other hand, we found it difficult to form a coherent strategy for dealing with quotes. Many times, an open quote would be found in one sentence and the closed quote would be found in an entirely different sentence. Therefore, we chose the simple strategy that quotes would anchor auxiliary trees that would adjoin to a neighboring sibling, namely, that sibling that was closer to the head sibling.

The Penn Treebank has an extensive list of empty elements which are used to define phenomena that are not usually expressed in LTAG. Among these are *U*, expressing a currency value, and *ICH*, indicating constituency relationships between discontinuous constituents. This observation led us to try two different strategies to cope with empty elements. The first strategy "ALL" is to include all empty elements in the grammar. The second strategy "SOME"

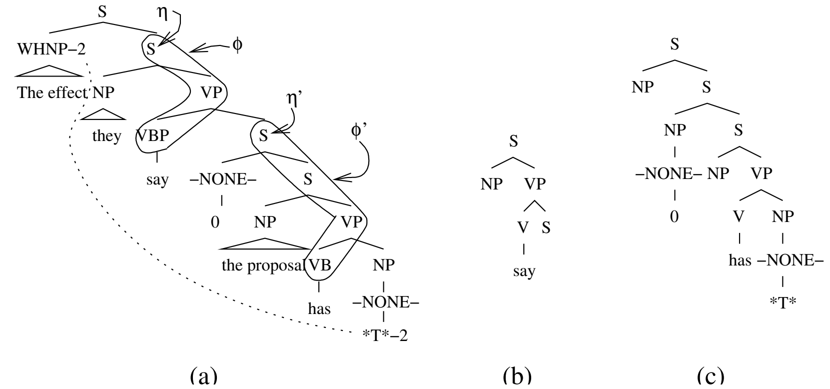

Figure 4.4. (a) Predicative auxiliary tree associated with ϕ is factored out because nodes η and η' have the same label, leaving trunk ϕ' to be extended to capture long distance movement. (b) The extracted predictive auxiliary tree. (c) The tree showing long distance movement.

is to only include empty elements demarcating empty subjects (0), empty PRO and passive NP trace (*), and traces (*T*) of syntactic movement; these are usually found in LTAG grammars of English.

The set of nonterminal and terminal labels in the Penn Treebank is extensive. A large set generally means that a greater number of trees are extracted from the Treebank; these trees could miss some generalizations and exacerbate the sparse data problem faced by any statistical model based on them. Also, some nonterminal labels are superfluous because they indicate structural configurations. For example, NX is used to label nodes in the internal structure of multi-word NP conjuncts inside an encompassing NP. If NX were replaced by NP, the tree extraction procedure can still determine that an instance of conjunction exists and take appropriate action. Conversely, distinctions that are made in a larger set of labels may aid the statistical model. For these reasons, we evaluated two different strategies. One strategy, "FULL," uses the original Penn Treebank label set. Another strategy, "MERGED," uses a reduced set of labels. In the latter approach, the original set is mapped onto a label set similar to that used in the XTAG grammar (XTAG-Group, 2001). In our approach, headedness and status as complement or adjunct was first determined according to the full set of labels before the trees were relabeled to the reduced set of labels.

Besides modifier auxiliary trees, there are predicative auxiliary trees which are generated as follows. During the bottom up extraction of trees, suppose trunk ϕ has a node η that shares the same label as another node η', where η' is a complement, not on ϕ, but is immediately dominated by a node on ϕ. In this case, a predicative auxiliary tree is extracted where η is its root, η' is its foot and with ϕ serving as its trunk. Subsequently, the path ϕ' dominated by η' becomes a candidate for being extended further. See Figure 4.4(a). This mechanism works in concert with other parts of our tree extraction procedure (notably complement and adjunct identification, merging of nonterminal labels – from SBAR to S, and policy of handling empty elements) in order to produce trees that localize long distance movement as shown in Figure 4.4(c).

Figure 4.4 also shows that the foot nodes of predicative auxiliary trees in LTAG typically correspond to sentential complements out of which extraction has occurred (XTAG-Group, 2001). In many cases, these are exactly the kinds of predicative auxiliary trees that the tree extraction procedure produces. On the other hand, there do exist anomalous cases. For example, it is possible for the procedure to produce a predicative auxiliary tree where the foot node corresponds to a sentential subject, even though in English it is not possible for extraction to occur from that position. One solution is to modify the tree extraction procedure to keep track explicitly of where extraction has occurred as trees are being constructed. Details of this solution are given in Chen (2001).

3. Evaluation

Each variation of tree extraction procedure was used to extract a grammar from Sections 02-21 of the Penn Treebank. These grammars were evaluated according to size, well formedness of trees, their coverage on Section 22, and their performance in supertagging Section 22. We subsequently evaluated truncated forms of these grammars which we term *cutoff* grammars.

Table 4.1. Size of various extracted grammars in number of tree frames and number of lexicalized trees

Comp Adjunct	*Empty Elements*	*Label Set*	*Grammar Size*	
			Frames	*Lexicalized Trees*
CA1	ALL	FULL	8996	118333
CA1	ALL	MERGED	5165	111220
CA1	SOME	FULL	8623	117527
CA1	SOME	MERGED	4911	110428
CA2	ALL	FULL	5354	116326
CA2	ALL	MERGED	2632	108370
CA2	SOME	FULL	4936	115335
CA2	SOME	MERGED	2366	107387

The grammars' sizes in terms of number of lexicalized trees and *tree frames* are shown in Table 4.1. Removing the anchor from a lexicalized tree yields a tree frame. In terms of different tree extraction strategies, MERGED yields more compact grammars than FULL, SOME more than ALL, and CA2 more

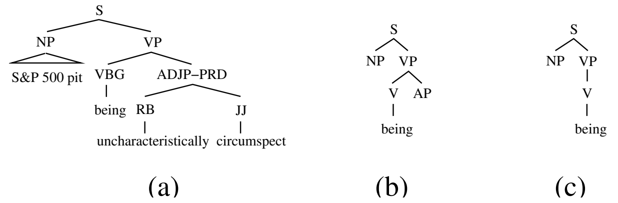

Figure 4.5. (a) Bracketed sentence S. (b) Lexicalized tree extracted from S using strategy CA1. (c) Lexicalized tree extracted from S using strategy CA2.

than CA1. Perhaps the last dichotomy requires more of an explanation. Basically, CA2 factors more nodes into auxiliary trees, with the result being that there are fewer trees because each one is structurally simpler.

We may also qualitatively judge grammars according to how well they satisfy our goal of extracting well formed trees in the sense of selecting the appropriate domain of locality and factoring recursion when necessary. There is not much difference between SOME and ALL because the empty elements that SOME ignores are the ones that are not usually captured by any LTAG grammar. Likewise, there is little difference between MERGED and FULL because most of MERGE's label simplification does not occur until after completion of tree extraction. The main difference lies between CA1 and CA2, strategies for labeling complements and adjuncts.

Nodes detected as complements of a particular lexical item belong in the same elementary tree, thus satisfying the criterion of localizing dependencies. We believe that CA1 labels nodes closer to what is traditionally found in LTAG grammars such as XTAG than does CA2, in that use of CA2 will generate less appropriate subcategorization frames because it tends to factor what might be considered as complements into separate auxiliary trees. It is difficult, however, to quantify the degree to which strategies CA1 and CA2 are successful in distinguishing complements from adjuncts because there are no precise definitions of these terms. Here we resign ourselves to a qualitative comparison of an example of a lexicalized tree extracted from the same sentence by a CA1 derived grammar G_1 (CA1-SOME-MERGED) and a CA2 derived grammar G_2 (CA2-SOME-MERGED). First, a tree frame F is selected from G_1 that is absent from G_2. A bracketed sentence S out of which the CA1 approach extracts F is then randomly culled from the training corpus. Figure 4.5(a) shows S, (b) shows the tree corresponding to the main verb extracted to G_1, (c) shows the analogous tree extracted to G_2. It is typical of the examples of divergence between CA1 and CA2 derived grammars: the CA1 approach leads to a verb subcategorization that is more complicated, yet more appropriate.

Table 4.2. Coverage of various extracted grammars and their corresponding supertagging performance. Coverage is in terms of percentage of tree frames and lexicalized trees in the test corpus. Missed coverage is divided into *in dict* – word seen in training corpus and *out dict* – word not seen in training corpus. Supertagging performance is based on either the full grammar or cutoff grammar, cutoff value $k = 3$.

Comp Adj	*Empty Elemt*	*Label Set*	*% found frames*	*% found lex trees*	*% miss in dict*	*% miss out dict*	*Supertag Acc*	*Cutoff Acc*
CA1	ALL	FULL	99.56	91.57	5.67	2.76	77.79	77.85
CA1	ALL	MERGED	99.82	92.18	5.06	2.76	78.70	78.57
CA1	SOME	FULL	99.60	91.66	5.58	2.76	78.00	78.07
CA1	SOME	MERGED	99.83	92.27	4.98	2.76	78.90	78.78
CA2	ALL	FULL	99.80	92.05	5.19	2.76	77.85	77.79
CA2	ALL	MERGED	99.94	92.71	4.53	2.76	78.25	78.25
CA2	SOME	FULL	99.83	92.14	5.10	2.76	78.07	78.08
CA2	SOME	MERGED	99.96	92.80	4.44	2.76	78.55	78.50

The various extracted grammars may also be evaluated according to breadth of coverage. In order to evaluate coverage of a particular grammar G, the strategy used to produce G was used to produce trees from held out data. We subsequently determined the degree of coverage of that strategy by the overlap in terms of tree frames and lexicalized trees as shown in Table 4.2. For lexicalized trees t extracted from held-out data such that $t \notin G$, we also show the percentage of cases where the *lexical anchors* of such trees t were or were not found in the training corpus (column *in dict* and *out dict* respectively). For example, the first row of Table 4.2 reports that use of strategy CA1-ALL-FULL resulted in a grammar such that 99.56% of instances of tree frames in held out data were available in the grammar, and 91.57% of instances of lexicalized trees in held out data were found in the grammar. Of the remaining 8.43%, 2.76% (*out dict*) of the lexicalized trees in the held out data involved words not seen in the training corpus. The remaining 5.67% therefore are anchored by words in the training corpus but the corresponding associations of tree frames and lexical items were not made in the training corpus. The table shows that strategies that reduce the number of extracted trees (SOME, CA2, MERGED) tend also to increase coverage.

We also measure the accuracies of supertagging models which are based on the various grammars that we are evaluating. Supertagging is the task of assigning the most appropriate lexicalized tree, or *supertag*, to each word of the input sentence (Bangalore and Joshi, 1999). In our experiments, we use the basic trigram model for supertagging, a linear time model that is quite akin to hidden Markov models for part of speech tagging. Results are shown in Table 4.2. Curiously, the grammars whose associated models achieved the highest ac-

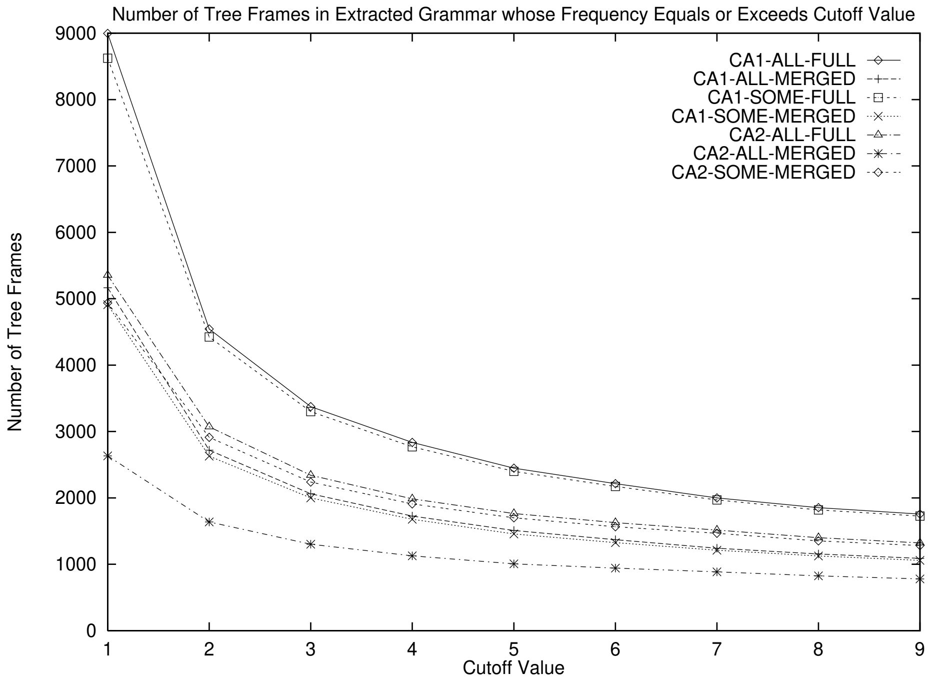

Figure 4.6. Number of tree frames occurring k times or more in the training corpus

curacies did not also have the highest coverage. For example, CA1-SOME-MERGED beat CA2-SOME-MERGED in terms of accuracy of supertagging model although the latter achieved higher coverage. This could possibly be caused by the fact that a high coverage grammar might have been obtained because it doesn't distinguish between contexts on which a statistical model can make distinctions. Alternatively, the cause may lie in the fact that a particular grammar makes better (linguistic) generalizations on which a statistical model can base more accurate predictions.

A large grammar may lead to a statistical model that is prohibitively expensive to run in terms of space and time resources. Furthermore, it is difficult to obtain accurate estimates of statistical parameters of trees with low counts. And, in any case, trees that only infrequently appear in the training corpus are also unlikely to appear in the test corpus. For these reasons, we considered the effect on a grammar if we removed those tree frames that occurred less than k times, for some cutoff value k. We call these *cutoff grammars*. As shown in Figure 4.6, even low values for k yield substantial decreases in grammar size.

Even though a cutoff grammar may be small in size, perhaps a statistical model based on such a grammar would degrade unacceptably in its accuracy.

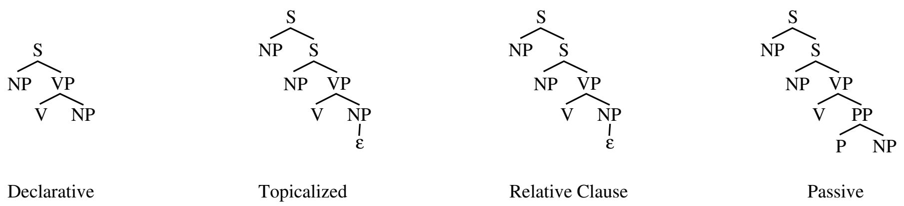

Figure 4.7. Some trees that are found in an XTAG tree family corresponding to transitive verbs

In order to see if this could indeed be the case, we trained and tested the supertagging model on various cutoff grammars. In the training data for these supertagging models, if a particular full grammar suggested a certain tree t_i for word w_i, but the cutoff grammar did not include t_i then word w_i was tagged as *miss*. The cutoff value of $k = 3$ was chosen in order to reduce the size of all of the original grammars at least in half. By the results shown in Table 4.2, it can be seen that use of a cutoff grammar instead of a full extracted grammar makes essentially no difference in the accuracy of the resulting supertagging model.

4. Extended Extracted Grammars

The grammars that have been produced with the tree extraction procedure suffer from sparse data problems as shown in Table 4.2 by the less than perfect coverage that these grammars achieve on the test corpus. This is perhaps one reason for the relatively low accuracies that supertagging models based on these grammars achieve compared to, for example, Srinivas (1997) and Chen et al. (1999). Many approaches may be investigated in order to improve the coverage. For example, although XTAG may be inadequate to entirely cover the Penn Treebank, it may be sufficient to ameliorate sparse data. Here we discuss how linguistic information as encoded in XTAG tree families may be used for this purpose. In particular, we use this information in order to augment an extracted grammar to obtain an *extended extracted grammar* and also deliver some preliminary results.

(XTAG-Group, 2001) explains that the tree frames anchored by verbs in the XTAG grammar are divided into tree families. Each tree family corresponds to a particular subcategorization frame. The trees in a given tree family correspond to various syntactic transformations as shown in Figure 4.7. Hence, if a word w_i is seen in the training corpus with a particular tree frame t_i, then it is likely for word w_i to appear with other tree frames $t \in T$ where T is the tree family to which t_i belongs.

This observation forms the basis of our experiment. The extracted grammar G_0 derived from CA1-SOME-MERGED was selected for this experiment. Call the extended grammar G_1. Initially, all of the trees $t \in G_0$ are inserted into G_1. Subsequently, for each lexicalized tree $t \in G_0$, lexicalized trees t' are added to G_1 such that t and t' share the same lexical anchor and the tree frames of t and t' belong to the same tree family. Out of approximately 60 XTAG tree families, those tree families that were considered in this experiment were those corresponding to relatively common subcategorization frames including intransitive, NP, PP, S, NP-NP, NP-PP and NP-S.

We achieved the following results. Recall that in Table 4.2 we divided the lapses in coverage of a particular extracted grammar into two categories: those cases where a word in the test corpus was not seen in the training corpus (*out dict*), and those cases where the word in the test corpus was seen in training, but not with the appropriate tree frame (*in dict*). Because our procedure deals only with reducing the latter kind of error, we report results from the latter's frame of reference. Using grammar G_0, the *in dict* lapse in coverage occurs 4.98% of the time whereas using grammar G_1, such lapses occur 4.61% of the time, an improvement of about 7.4%. This improvement must be balanced against the increase in grammar size. Grammar G_0 has 4900 tree frames and 114850 lexicalized trees. In comparison, grammar G_1 has 4999 tree frames and 372262 lexicalized trees.

The results are quite encouraging and there exist avenues for improvement. For example, the number of lexicalized trees in the extended extracted grammar can be reduced if we account for morphological information. Note for instance that "drove," the simple past tense form of the verb "to drive," does not license passive structures. Instead of capitalizing on this distinction, our current procedure simply associates all of the tree frames in the transitive tree family with "drove." Also, in order to use these extended extracted grammars in statistical models, there needs to be a way to estimate parameters for lexicalized trees that are unseen in the training corpus.

In related work, we have conducted experiments to quantitatively extract a measure of similarity between pairs of supertags (tree frames) by taking into account the distribution of the supertags with words that anchor them Chen (2001). When a particular supertag-word combination is absent from the training corpus, instead of assigning it a zero probability, we assign a probability that is obtained by considering similar supertags and their probability of being assigned with this word. This method seems to give promising results, circumventing the need for manually designed heuristics such as those found in the supertagging work of Srinivas (1997) and Chen et al. (1999). In applying this strategy to our extracted grammar, we show that significant improvements in supertagging accuracy can be obtained.

5. Related Work

Neumann (1998) presents a method for extracting an LTAG from a treebank. Like our work, that method determines the trunks of elementary trees by finding paths in the tree with the same headword, headwords being determined by a head percolation table. Unlike our work, that method factors neither adjuncts nor instances of conjunction into auxiliary trees. As a result, it generates many more trees than we do. Using only Sections 02 through 04 of the Penn Treebank, it produces about 12000 tree frames. Our approaches produces about 2000 to 9000 tree frames using Sections 02-21 of the Penn Treebank.

Xia (1999) also presents work in extracting an LTAG from a treebank, work that was done in parallel with our own work. Like our work, that method determines the trunk of elementary trees by finding paths in the tree with the same headword. Furthermore, it factors adjuncts (according to CA2 only) into separate auxiliary trees. Also, following our suggestion it factors instances of conjunction into separate auxiliary trees. Xia's approach yields either 3000 or 6000 tree frames using Sections 02-21 of the Penn Treebank, depending on the preterminal tagset used. Our work explores a wider variety of parameters in the extraction of trees, yielding grammars that have between about 2000 to 9000 tree frames on the same training data. Unlike our work in the extraction of an LTAG, Xia's method extracts a multi-component LTAG through coindexation of traces in the Penn Treebank. Another difference is that it is more concerned with extraction of grammar for the purpose of grammar development (for which it for example makes the distinction between extracted tree frames that are grammatically correct and those that are incorrect), whereas our current work in extraction of grammar betrays our ultimate interest in developing statistical models for parsing (for which we perform an investigation of coverage, supertagging accuracy, effect of cutoff frequency, as well as explore the issue of extending extracted grammars using XTAG tree families for eventual use in statistical smoothing).

Chiang (2000) presents an alternative method for extracting a lexicalized grammar from a treebank. Like our work, that method uses a head percolation table in order to find the trunks of elementary trees, factors adjuncts into separate auxiliary trees, and factors conjuncts into separate auxiliary trees. There also exist various differences. Instead of extracting an LTAG, that method extracts a lexicalized tree insertion grammar (LTIG) (Schabes, 1993). Certain configurations of dependencies that are legal in LTAG are not easily represented in LTIG. Another difference is the extracted LTIG of Chiang (2000) does not represent any empty elements. Insofar as empty elements are linguistically merited, the extracted LTIG may be said to be linguistically impoverished. On the other hand, not including empty elements is one reason why the extracted LTIG is so compact (3600 tree frames), using the label set FULL, when compared to the our smallest comparable grammar, CA2-SOME-FULL (4936 tree frames). Another difference is that work documents the extraction of only one kind of grammar whereas we experiment with different parameterizations of extracted grammar. However, it does include results of the additional task of experimenting with statistical parsing using the extracted LTIG, showing that its performance is quite close to parsing models based on a lexicalized context free formalism (Collins, 1997; Charniak, 2000). That work also experiments with smoothing probability distributions based on the extracted LTIG, similar to our motivation behind extended extracted grammars, but only by backing off from tree frames to their corresponding preterminals (parts of speech).

6. Conclusions

Our work presents new directions in both the extraction of an LTAG from the Penn Treebank as well as its application to statistical models. In the extraction of an LTAG from the Penn Treebank, we have extended Neumann's procedure to produce less unwieldy grammars by factoring recursion that is found in adjuncts as well as in instances of conjunction. We have explored the effects that different definitions of complement and adjunct, whether or not to ignore empty elements, and the amount of detail in label sets have on the quality and size of the extracted grammar, as well as ability to cover an unseen test corpus. We have also evaluated those grammars according to supertagging accuracy. We have experimented with the notion of cutoff grammar, and seen that these grammars are more compact and yet yield little in the way of supertagging accuracy. We have introduced the notion of an extended extracted grammar, which enriches an extracted grammar by use of an external resource, namely, the tree families of XTAG. We have seen that this technique expands an extracted grammar's coverage, and discussed how this technique may be developed in order to achieve better results.

There are a number of ways to extend this work of extracting an LTAG from the Penn Treebank. Because our primary goal is to develop a grammar around which to base statistical models of parsing, we are in particular interested in better procedures for coping with the sparse data problems engendered by the relatively large sizes of extracted grammars. Extended extracted grammars provide one means of doing this, albeit it is an incomplete solution because for it to be used in a statistical framework, there also needs to be a method for estimating parameters for those lexicalized trees that are unseen in the training corpus, but are part of the extended extracted grammar. More complete solutions can be found in Chen (2001), which provides a comparative study of several different approaches for smoothing an extracted grammar. Subsequently, we would like to see how the extracted grammar could be used as a basis for probabilistic models for LTAG, as delineated in Schabes (1992) and Resnik (1992). These models can then be applied to parsing, as in Chiang (2000). The work of Chiang (2000) may be extended to determine how different parameterizations of extracted grammars affect parsing accuracy, as well as to rate different approaches to creating extended extracted grammars. Models that are based on extracted grammars can also be potentially useful in applications other than parsing, such as stochastic natural language generation, as in Bangalore et al. (2001) and Chen et al. (2002).

Acknowledgments

This work was supported by NSF grants #SBR-9710411 and #GER-9354869.

References

Bangalore, S., Chen, J., and Rambow, O. (2001). Impact of quality and quantity of corpora on stochastic generation. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, PA.

Bangalore, S. and Joshi, A. K. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

Charniak, E. (1996). Tree-bank grammars. Technical Report CS-96-02, Brown University, Providence, RI.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of First Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, Seattle, WA.

Chen, J. (2001). *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information*. PhD thesis, University of Delaware.

Chen, J., Bangalore, S., Rambow, O., and Walker, M. (2002). Towards automatic generation of natural language generation systems. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan.

Chen, J., Bangalore, S., and Vijay-Shanker, K. (1999). New models for improving supertag disambiguation. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway.

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

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*.

Collins, M. (1997). Three generative lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*.

Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33th Annual Meeting of the Association for Computational Linguistics*.

Marcus, M., Kim, G., Mary, Marcinkiewicz, MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn treebank: Annotating predicate argument structure. In *Proceedings of the 1994 Human Language Technology Workshop*, pages 110–115.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Neumann, G. (1998). Automatic extraction of stochastic lexicalized tree grammars from treebanks. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 120–123.

Resnik, P. (1992). Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 92)*, pages 418–424, Copenhagen, Denmark.

Schabes, Y. (1992). Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 92)*, pages 426–432, Copenhagen, Denmark.

Schabes, Y. (1993). Lexicalized context-free grammars. Technical Report 93-01, Mitsubishi Electric Research Laboratories, Cambridge, MA.

Schabes, Y., Abeillé, A., and Joshi, A. K. (1988). Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary.

Srinivas, B. (1997). Performance evaluation of supertagging for partial parsing. In *Proceedings of the Fifth International Workshop on Parsing Technologies*, pages 187–198, Cambridge, MA.

Xia, F. (1999). Extracting tree adjoining grammars from bracketed corpora. In *Fifth Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.

XTAG-Group, T. (2001). A Lexicalized Tree Adjoining Grammar for English. Technical report, University of Pennsylvania. Updated version available at http://www.cis.upenn.edu/˜xtag.

Index