14

Tree Adjoining Grammars and their Application to Statistical Parsing

ARAVIND JOSHI AND ANOOP SARKAR

14.1 Introduction

Tree-Adjoining Grammars (TAG) represent a tree ‘manipulating’ system. Viewed generatively, it is a tree rewriting system and viewed analytically, it leads to a parser, which parses a sentence into its component elementary trees (lexically anchored in the case of a Lexicalized Tree-Adjoining Grammar (LTAG)) and the history of composition of these trees by the operations of substitution and adjoining.

Large scale LTAG grammars have been constructed by hand for English at UPenn (XTAG Group 2001) and French (at the TALANA group, University of Paris 7, France) and somewhat smaller ones for German (at DFKI, Saarbrücken, Germany), Korean (at UPenn), Chinese (at UPenn), and Hindi (at CDAC, Pune, India). The earliest stochastic variant of TAG was proposed by (Resnik 1992, Schabes 1992). LTAG grammars have been extracted from annotated corpora (Xia et al. 2001, Xia 2001, Chiang 2000, Chen and Vijay-Shanker 2000), which in turn have been used for lexicalized statistical parsing (Chiang 2000, Sarkar 2001). The statistical parsing work done in TAGs emphasizes the use of lexicalized elementary trees and the recovery of the best derivation for a given sentence rather than the best parse tree. We believe that these aspects of LTAG make them relevant to readers of this volume.

The plan of this paper is as follows. In Section 14.2 we will present a short introduction to LTAG, pointing out specifically how LTAG arises in the natural process of lexicalization of context-free grammars (CFG).

Data Oriented Parsing.
Rens Bod, Remko Scha and Khalil Sima'an (eds.).
Copyright © 2002, CSLI Publications.

The resulting system is however, more powerful than CFGs, both in terms of weak generative capacity (string sets) and strong generative capacity (in terms of structural descriptions associated with the strings), the latter property is more relevant in the context of this volume. In Sections 14.3 and 14.4 we will describe the stochastic models for TAGs and LTAGs and their application to statistical parsing. In Section 14.5 we show how a well-formed probabilistic generative process can be defined for stochastic TAGs. Finally, in Section 14.6 we will discuss some recent relevant work in the context of this volume.

14.2 Tree-adjoining grammars

Tree-adjoining grammar (TAG) is a formal tree rewriting system. TAG and Lexicalized Tree-Adjoining Grammar (LTAG) have been extensively studied both with respect to their formal properties and to their linguistic relevance. TAG and LTAG are formally equivalent, however, from the linguistic perspective LTAG is the system we will be concerned with in this paper. We will often use these terms TAG and LTAG interchangeably.

The motivations for the study of LTAG are both linguistic and formal. The elementary objects manipulated by LTAG are structured objects (trees or directed acyclic graphs) and not strings. Using structured objects as the elementary objects of the formal system, it is possible to construct formalisms whose properties relate directly to the study of strong generative capacity (i.e., structural descriptions), which is more relevant to the linguistic descriptions than the weak generative capacity (sets of strings).

Each grammar formalism specifies a domain of locality, i.e., a domain over which various dependencies (syntactic and semantic) can be specified. It turns out that the various properties of a formalism (syntactic, semantic, computational, and even psycholinguistic) follow, to a large extent, from the initial specification of the domain of locality.

14.2.1 Domain of locality of CFGs

In a context-free grammar (CFG) the domain of locality is the one level tree corresponding to a rule in a CFG (Figure 68). It is easily seen that the arguments of a predicate (for example, the two arguments of *likes*) are not in the same local domain. The two arguments are distributed over the two rules (two domains of locality)– $S \rightarrow NP\, VP$ and $VP \rightarrow V\ NP$. They can be brought together by introducing a rule $S \rightarrow NP\ V\ VP$. However, then the structure provided by the VP node is lost. We should also note here that not every rule (domain) in the CFG in (Figure 68) is lexicalized. The four rules on the right are lexicalized, i.e., they have

CFG G	S $\rightarrow$ NP VP	NP $\rightarrow$ Harry	
	VP $\rightarrow$ V NP	NP $\rightarrow$ peanuts	
	VP $\rightarrow$ VP ADV	V $\rightarrow$ likes	
		ADV $\rightarrow$ passionately	

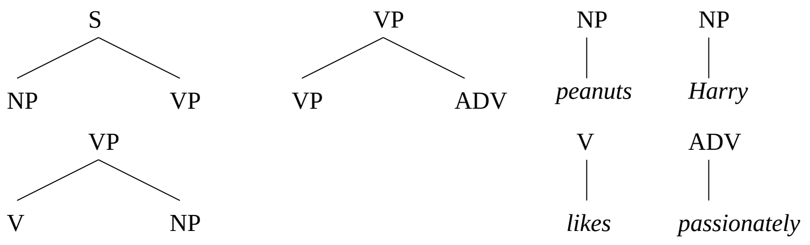


FIGURE 68 Domain of locality of a context-free grammar

a lexical anchor. The rules on the left are not lexicalized. The second and the third rules on the left are almost lexicalized, in the sense that they each have a preterminal category (V in the second rule and ADV in the third rule), i.e., by replacing V by *likes* and ADV by *passionately* these two rules will become lexicalized. However, the first rule on the left ($S \rightarrow NP\ VP$) cannot be lexicalized. Can a CFG be lexicalized, i.e., given a CFG, G, can we construct another CFG, G', such that every rule in G' is lexicalized and $T(G)$, the set of (sentential) trees (i.e., the tree language of G) is the same as the tree language $T(G')$ of G'? It can be shown that this is not the case (Joshi and Schabes 1997). Of course, if we require that only the string languages of G and G' be the same (i.e., they are weakly equivalent) then any CFG can be lexicalized. This follows from the fact that any CFG can be put in the Greibach normal form where each rule is of the form $A \rightarrow w\ B1\ B2\ ...\ Bn$ where w is a lexical item and the $B's$ are nonterminals. The lexicalization we are interested in requires the tree languages (i.e., the set of structural descriptions) be the same, i.e., we are interested in 'strong' lexicalization. To summarize, a CFG cannot be strongly lexicalized by a CFG. This follows from the fact that the domain of locality of CFG is a one level tree corresponding to a rule in the grammar. Note that there are two issues we are concerned with here– lexicalization of each elementary domain and the encapsulation of the arguments of the lexical anchor in the elementary domain of locality. The second issue is independent of the first issue. From the mathematical point of view the first issue, i.e., the lexicalization of the elementary domains of locality is the crucial one. We can obtain strong lexicalization without satisfying the requirement specified in the second issue (encapsulation of the arguments of the lexical anchor). Of course, from the linguistic point of view the second issue is very crucial. What this means is that among all possible strong lexicalizations we should choose only those that meet the requirements of the second issue. For our discussions in this paper we will assume that we always make such a choice.

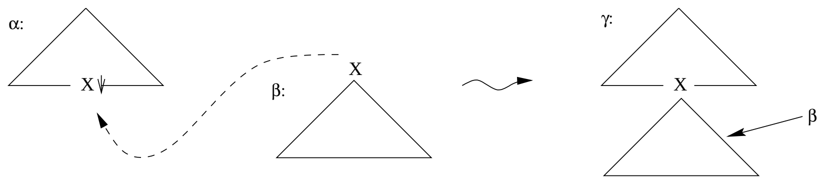


FIGURE 69 Substitution

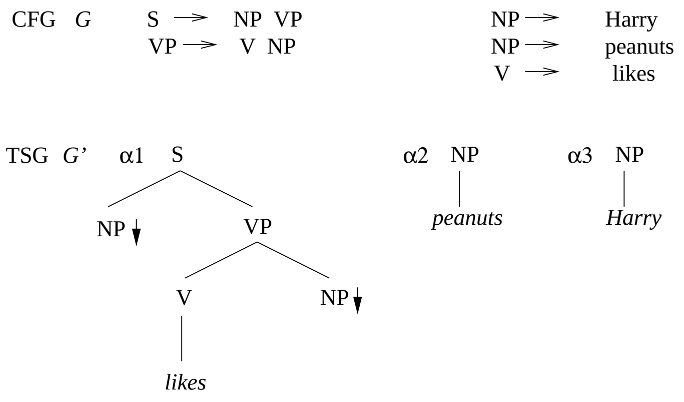


FIGURE 70 Tree substitution grammar

14.2.2 Lexicalization of CFGs

Now we can ask the following question. Can we strongly lexicalize a CFG by a grammar with a larger domain of locality? Figure 69 and Figure 70 show a tree substitution grammar where the elementary objects (building blocks) are the three trees in Figure 70 and the combining operation is the tree substitution operation shown in Figure 69. Note that each tree in the tree substitution grammar (TSG), G' is lexicalized, i.e., it has a lexical anchor. It is easily seen that G' indeed strongly lexicalizes G. However, TSG's fail to strongly lexicalize CFG's in general. We show this by an example. Consider the CFG, G, in Figure 71 and a proposed TSG, G'. It is easily seen that although G and G' are weakly equivalent they are not strongly equivalent. In G', suppose we start with the tree α_1 then by repeated substitutions of trees in G' (a node marked with a vertical arrow denotes a substitution site) we can grow the right side of α_1 as much as we want but we cannot grow the left side. Similarly for α_2 we can grow the left side as much as we want but not the right side. However, trees in G can grow on both sides. Hence, the TSG, G', cannot strongly lexicalize the CFG, G (Joshi and Schabes 1997).

CFG G S $\rightarrow$ S S (non-lexical)
             S $\rightarrow$ a (lexical)

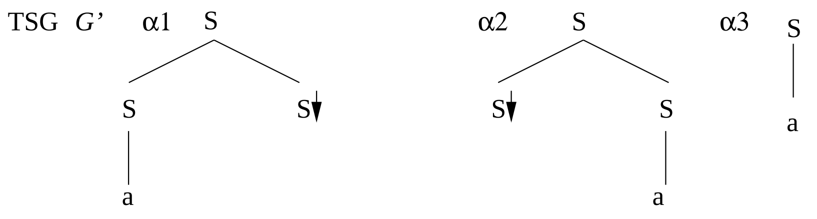


FIGURE 71 A tree substitution grammar

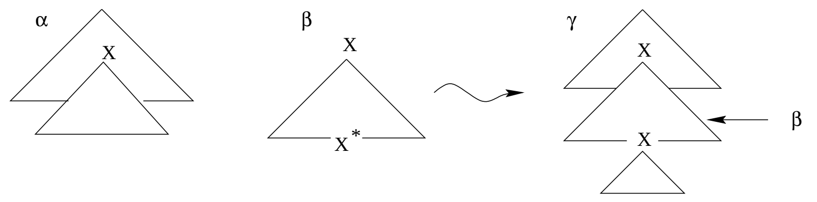


FIGURE 72 Adjoining

We now introduce a new operation called 'adjoining' as shown in Figure 72. Adjoining involves splicing (inserting) one tree into another. More specifically, a tree β as shown in Figure 72 is inserted (adjoined) into the tree α at the node X resulting in the tree γ. The tree β, called an **auxiliary tree**, has a special form. The root node is labeled with a nonterminal, say X and on the frontier there is also a node labeled X called the foot node (marked with *). There could be other nodes (ter-

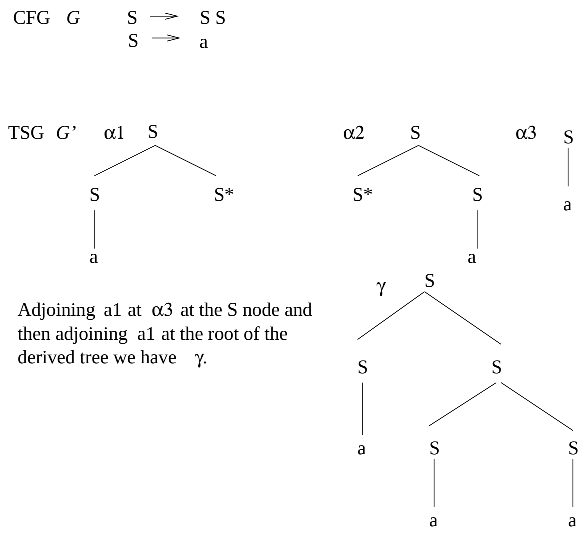

CFG G
S → S S
S → a
TSG G'
α1
α2
α3
S
S*
a
γ
Adjoining a1 at α3 at the S node and then adjoining a1 at the root of the derived tree we have γ.

FIGURE 73 Adjoining arises out of lexicalization

minal or nonterminal) nodes on the frontier of β, the nonterminal nodes will be marked as substitution sites (with a vertical arrow). Thus if there is another occurrence of X (other than the foot node marked with *) on the frontier of β it will be marked with the vertical arrow and that will be a substitution site. Given this specification, adjoining β to α at the node X in α is uniquely defined. Adjoining can also be seen as a pair of substitutions as follows: The subtree at X in α is detached, β is substituted at X and the detached subtree is then substituted at the foot node of β. A tree substitution grammar when augmented with the adjoining operation is called a tree-adjoining grammar (lexicalized tree-adjoining grammar because each elementary tree is lexically anchored). In short, LTAG consists of a finite set of elementary trees, each lexicalized with at least one lexical anchor. The elementary trees are either initial or auxiliary trees. Auxiliary trees have been defined already. Initial trees are those for which all nonterminal nodes on the frontier are substitution nodes. It can be shown that any CFG can be strongly lexicalized by an LTAG (Joshi and Schabes 1997).

In Figure 73 we show a TSG, G', augmented by the operation of adjoining, which strongly lexicalizes the CFG, G. Note that the LTAG looks the same as the TSG considered in Figure 71. However, now trees α_1 and α_2 are auxiliary trees (marked with *) that can participate in adjoining. Since adjoining can insert a tree in the interior of another tree it is possible to grow both sides of the tree α_1 and tree α_2, which was not possible earlier with substitution alone. In summary, we have shown that by increasing the domain of locality we have achieved the following: (1) lexicalized each elementary domain, (2) introduced an operation of adjoining, which would not be possible without the increased domain of locality (note that with one level trees as elementary domains adjoining becomes the same as substitution since there are no interior nodes to be operated upon), and (3) achieved strong lexicalization of CFGs.

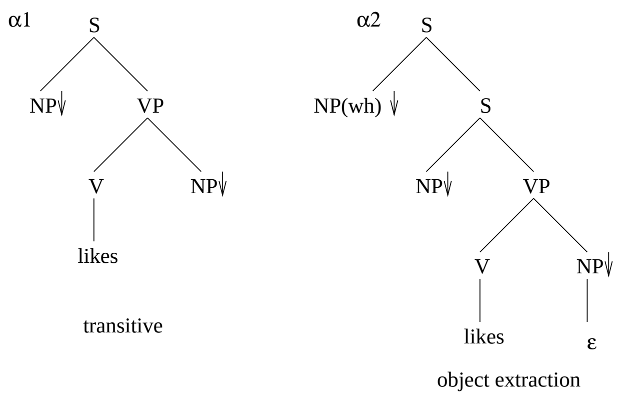


FIGURE 74 LTAG: Elementary trees for *likes*

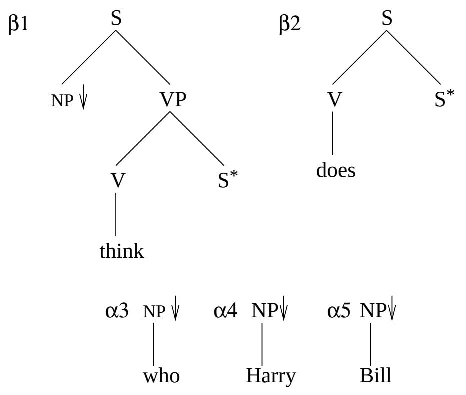


FIGURE 75 LTAG: Sample elementary trees

14.2.3 Lexicalized tree-adjoining grammars

Rather than giving formal definitions for LTAG and derivations in LTAG we will give a simple example to illustrate some key aspects of LTAG. We show some elementary trees of a toy LTAG grammar of English. Figure 74 shows two elementary trees for a verb such as *likes*. The tree α_1 is anchored on *likes* and encapsulates the two arguments of the verb. The tree α_2 corresponds to the object extraction construction. Since we need to encapsulate all the arguments of the verb in each elementary tree for *likes*, for the object extraction construction, for example, we need to make the elementary tree associated with *likes* large enough so that the extracted argument is in the same elementary domain. Thus, in principle, for each 'minimal' construction in which *likes* can appear (for example, subject extraction, topicalization, subject relative, object relative, passive, etc.) there will be an elementary tree associated with that construction. By 'minimal' we mean when all recursion has been factored away. This factoring of recursion away from the domain over which the dependencies have to be specified is a crucial aspect of LTAGs as they are used in linguistic descriptions. This factoring allows all dependencies to be localized in the elementary domains. In this sense, there will, therefore, be no long distance dependencies as such. They will all be local and will become long distance on account of the composition operations, especially adjoining.

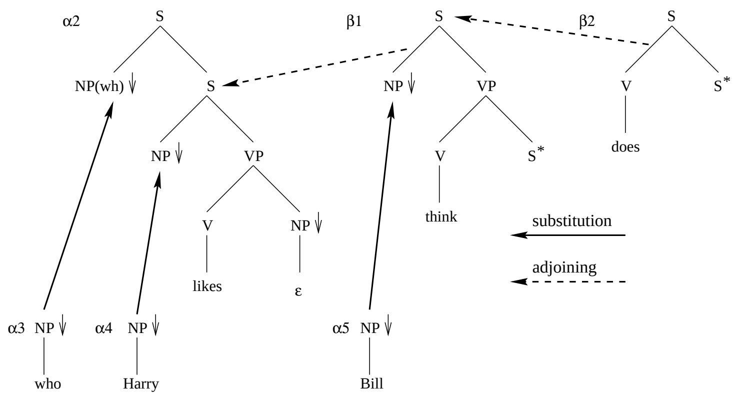


FIGURE 76 LTAG derivation for *who does Bill think Harry likes*

Figure 75 shows some additional trees. Trees α_3, α_4, and α_5 are initial trees and trees β_1 and β_2 are auxiliary trees with foot nodes marked with *. A derivation using the trees in Figure 74 and Figure 75 is shown in

Figure 76. The trees for *who* and *Harry* are substituted in the tree for *likes* at the respective *NP* nodes, the tree for *Bill* is substituted in the tree for *think* at the *NP* node, the tree for *does* is adjoined to the root node of the tree for *think* tree (adjoining at the root node is a special case of adjoining), and finally the derived auxiliary tree (after adjoining β_2 to β_1) is adjoined to the indicated interior S node of the tree α_2. This derivation results in the **derived tree** for *who does Bill think Harry likes* as shown in Figure 77. Note that the dependency between *who* and the complement *NP* in α_2 (local to that tree) has been stretched in the derived tree in Figure 77. This tree is the conventional tree associated with the sentence.

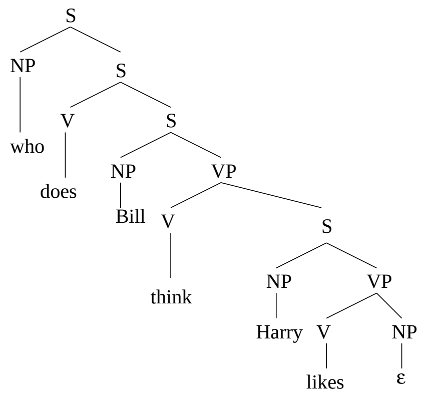


FIGURE 77 LTAG derived tree for *who does Bill think Harry likes*

However, in LTAG there is also a derivation tree, the tree that records the history of composition of the elementary trees associated with the lexical items in the sentence. This derivation tree is shown in Figure 78. The nodes of the tree are labeled by the tree labels such as α_2 together with the lexical anchor.[73] The derivation tree is the crucial derivation structure for LTAG. We can obviously build the derived tree from the derivation tree. For semantic computation the derivation tree (and not the derived tree) is the crucial object. Compositional semantics is defined on the derivation tree. The idea is that for each elementary tree there is a semantic representation associated with it and these representations are composed using the derivation tree. Since the semantic representation for

[73]The derivation trees of LTAG have a close relationship to the dependency trees, although there are some crucial differences; however, the semantic dependencies are the same. See (Rambow and Joshi 1995) for more details.

each elementary tree is directly associated with the tree there is no need to reproduce necessarily the internal hierarchy in the elementary tree in the semantic representation (Joshi and Vijay-Shanker 1999). This allows the so-called 'flat' semantic representation as well as helps in dealing with some non-compositional aspects as in the case of rigid and flexible idioms.

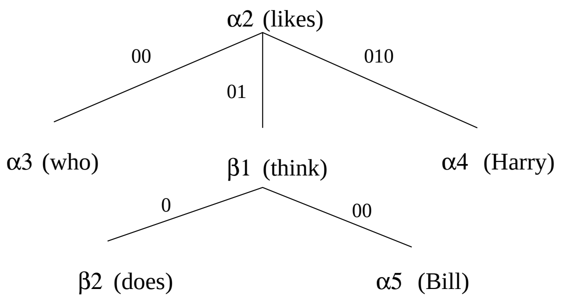


FIGURE 78 LTAG derivation tree

14.2.4 Some important properties of LTAG

The two key properties of LTAG are (1) extended domain of locality (EDL) (for example, as compared to CFG), which allows (2) factoring recursion from the domain of dependencies (FRD), thus making all dependencies local. All other properties of LTAG (mathematical, linguistic, and even psycholinguistic) follow from EDL and FRD. TAGs (LTAGs) belong to the so-called class of mildly context-sensitive grammars (Joshi 1985). Context-free languages (CFL) are properly contained in the class of languages of LTAG, which in turn are properly contained in the class of context-sensitive languages. There is a machine characterization of TAG (LTAG), called embedded pushdown automaton (EPDA) (Vijay-Shanker 1987), i.e., for every TAG language there is an EPDA which corresponds to this (and only this) language and the language accepted by any EPDA is a TAG language. EPDAs have been used to model some psycholinguistic phenomena, for example, processing crossed dependencies and nested dependencies have been discussed in (Joshi 1990). With respect to formal properties, the class of TAG languages enjoys all the important properties of CFLs, including polynomial parsing (with complexity $O(n^6)$).

Large scale wide coverage grammars have been built using LTAG, the XTAG system (LTAG grammar and lexicon for English and a parser) being the largest so far (for further details see (XTAG Group 2001). In

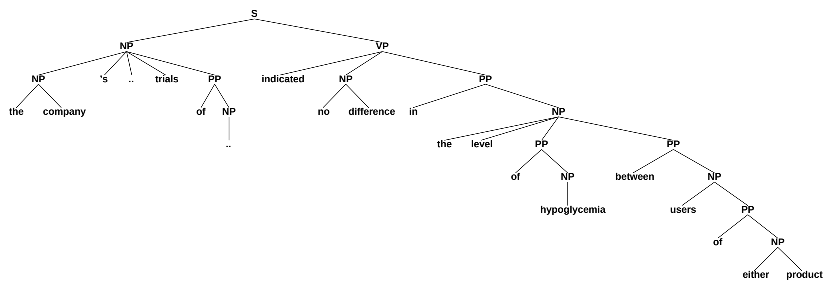


FIGURE 79 Parse tree for the sentence: *the company 's clinical trials of both its animal and human-based insulins indicated no difference in the level of hypoglycemia between users of either product*

the XTAG system, each node in each LTAG tree is decorated with two feature structures (top and bottom feature structures), in contrast to the CFG based feature structure grammars. This is necessary because adjoining can augment a tree internally, while in a CFG based grammar or even in a tree substitution grammar a tree can be augmented only at the frontier. It is possible to define adjoining and substitution (as it is done in the XTAG system) in terms of appropriate unifications of the top and bottom feature structures. Because of FRD (factoring recursion from the domain of dependencies), there is no recursion in the feature structures. Therefore, in principle, feature structures can be eliminated. However, they are crucial for linguistic descriptions. Constraints on substitution and adjoining are modeled via these feature structures (Vijay-Shanker 1987). This method of manipulating feature structures is a direct consequence of the extended domain of locality of LTAG.

14.3 Statistical Parsing with Tree Adjoining Grammars

Before we provide a formal definition for stochastic Tree Adjoining Grammars, we provide some motivation for its use in statistical parsing. In building a statistical parser our task is to find the sub-parts or constituents for naturally occuring sentences are shown in Figure 79.

These constituents are recursively embedded within one another and hence we decompose the entire structure assigned to a sentence into smaller parts. A common theory underlying this decomposition is that of context-free grammars (CFGs). For example, for the constituency structure given in Figure 79, we extract context-free rules of the kind

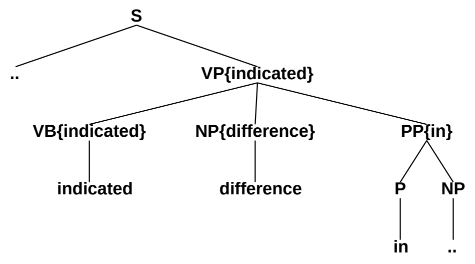


FIGURE 80 Context-free rule or a tree of depth= 1.

shown in Figure 14.3. In order for the phrase structure to be sensitive to the lexical information each constituent has a *head* word that represents it in the phrase structure.

However, the right hand side of each context-free rule has multiple constituent elements each with its own head word. Therefore, rules can be lexicalized with several words at once. Since words occur sparsely in any corpus, we are quite unlikely to see three or more words that have occurred together in the training data again in the test data. For this reason, standard CFG-based statistical parser impose a further independence assumption that the right hand side of each rule is produced by a Markov process. The generation of each CFG rule is shown in Figure 81. Each dependent of the head of the VP phrase which in this case is the verb *indicated* is generated independently of the other dependents on the head. For example, the NP headed by *difference* and the PP headed by *in* are attached to the verb independently of each other. There is an added symbol $STOP$ to make well-defined 0-th order Markov process.

However, the independence assumptions underlying the decomposition shown in Figure 81 are violated in the corpus. Let us consider subtrees of depth greater than 1 and their empirical distribution in the Penn Treebank WSJ corpus. In subtrees where a VP node dominates another VP node as shown in Figure 82 we see a marked difference in the expected attachment of a PP to the various VP positions. The PP is far more likely to attach the lower VP rather than to the higher VP. This additional fact about the distribution of subtrees of phrase structure has to be added into a model which makes the kind of independence assumptions shown in Figure 81. Figure 82 also shows that the independence assumptions discussed earlier can be very beneficial. Ignoring an

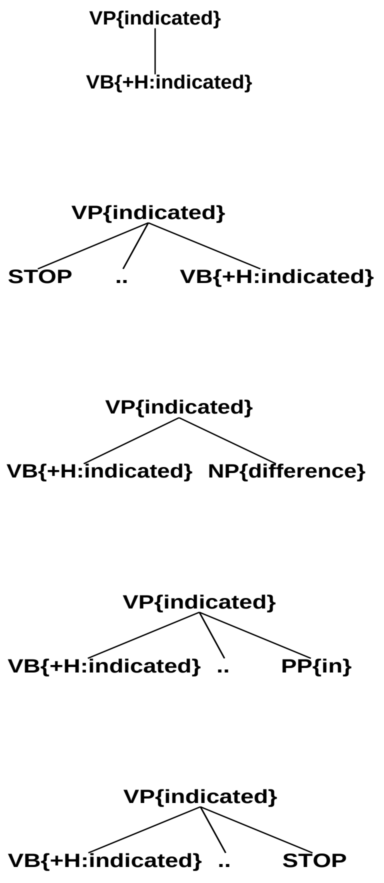


FIGURE 81 Independence assumptions that decompose the right hand side of the CFG rule.

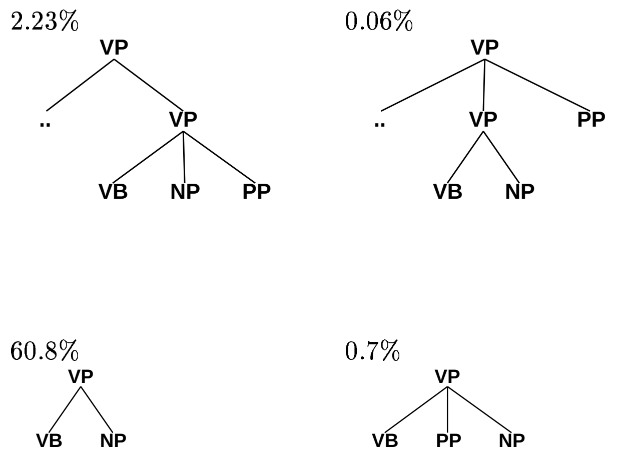


FIGURE 82 Independence assumptions are violated by only considering trees of depth= 1.

optional PP can help us generalize to new cases in the test data even if we have not seen a particular right hand side of a CFG rule used in the test data.

The question is whether we can retain the advantages of the independence assumptions shown in Figure 81 while at the same time being sensitive to the empirical facts about subtrees shown in Figure 82.

CFG rules can be viewed as trees of depth 1. A formalism that uses trees of depth greater than 1 would be able to capture the facts since it will be more sensitive to various possible attachment points along the spine of the tree. TAG is one such formalism.

Constructing a derivation in LTAG proceeds as follows: each word in a sentence is assigned a set of trees. Each of these trees assigned to the words in the sentence can combine with each other to produce a derivation from which an ordinary constituency phrase structure tree is produced. The yield of this tree is the input sentence.

Hence, TAG is an alternative method to the modeling of bilexical dependencies. For example, see Figure 83.

Each combination of a pair of trees is given a probability. For example, Figure 84 shows the probability model for substituting a tree for a non-terminal at the frontier of another tree.

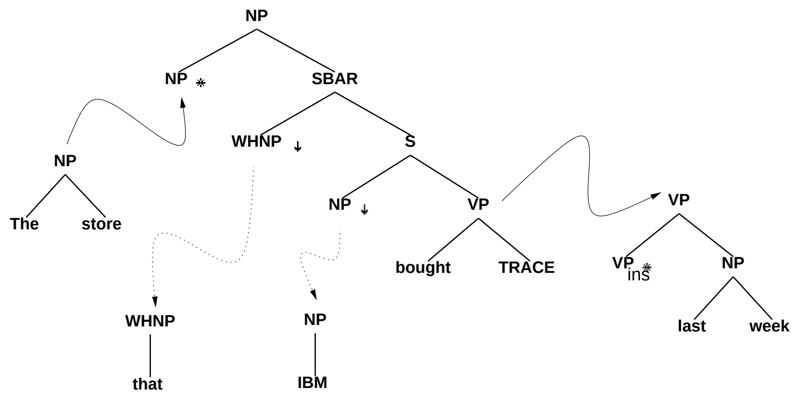
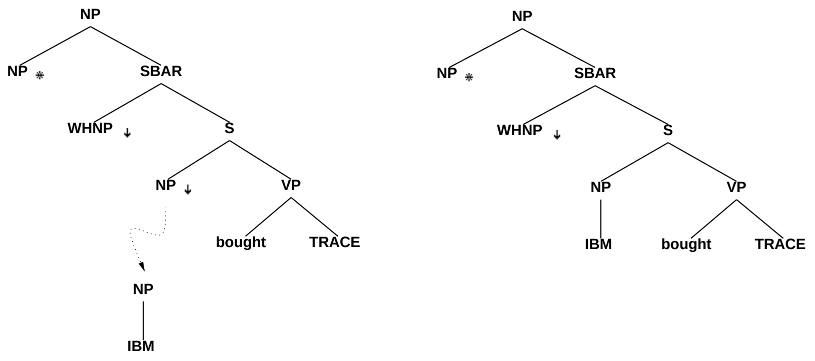
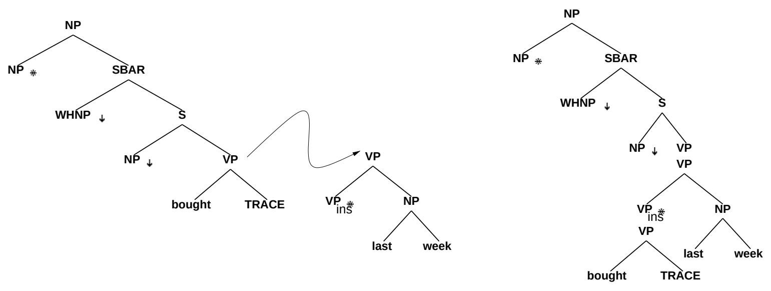


FIGURE 83 Alternative modeling of bilexical dependencies using a stochastic TAG.



$$\sum_{t'} \mathcal{P}(t, \eta \to t') = 1$$

FIGURE 84 TAG: Substitution



$$\mathcal{P}(t, \eta \rightarrow NA) + \sum_{t'} \mathcal{P}(t, \eta \rightarrow t') = 1$$

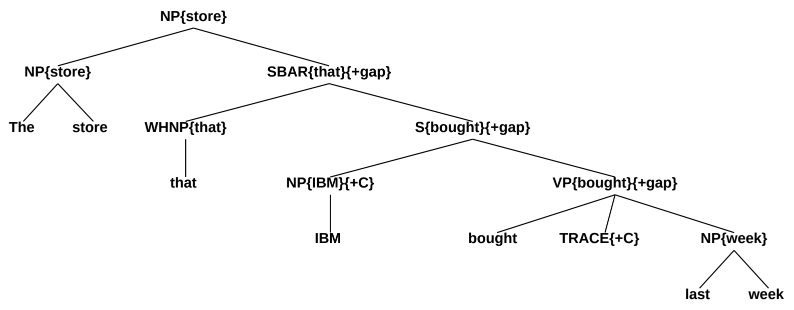
FIGURE 85 TAG: Adjunction

Non-terminals that occur internal to each tree can also be rewritten by the operation of adjunction which replaces the non-terminal with a tree. For example, the adjunction of a tree into a node η is shown in Figure 85. The additional term that is missing from the substitution model is the probability that the non-terminal is not re-written during the derivation (called the null-adjunction or NA probability).

TAGs have the following attractive properties as a framework for statistical parsing:

- Locality and independence assumptions are captured elegantly.
- Simple and well-defined probability model.
- Parsing can be treated in two steps:
  1. Classification: structured labels (elementary trees) are assigned to each word in the sentence.
  2. Attachment: the elementary trees are connected to each other to form the parse.

In addition, statistical parsers that are serious about interacting with a component that links the sentence to a meaning have to produce more than the phrase structure of each sentence. A more embellished parse in which phenomena such as predicate-argument structure, subcategorization and movement are given a probabilistic treatment is often expected from a parser. A CFG parser has to deal with such extensions by appealing to some kind of 'feature' based account as shown in Figure 86. Note that such additional information already forms part of every lexicalized tree in the TAG framework as shown in Figure 83.



NP	→	[	NP{+H}	SBAR{+gap}		]
SBAR{+gap}	→	[	WHNP	S{+H}{+C}{+gap}		]
S{+gap}	→	[	NP{+C}	SBAR{+H}{+gap}		]
VP{+gap}	→	[	VB{+H}	TRACE{+C}	NP	]

FIGURE 86 Features typically used in a Bilexical CFG parser (see Collins 1999)

14.4 Stochastic Tree Adjoining Grammars

(Resnik 1992) provided some early motivation for a stochastic version of Tree Adjoining Grammars and gave a formal definition of stochastic TAG. Simultaneously, (Schabes 1992) also provided an identical stochastic version of TAG and also extended the Inside-Outside algorithm for CFGs (Lari and Young 1990) to stochastic TAGs. (Schabes 1992) also performed experiments to show that a stochastic TAG can be learnt from the ATIS corpus.

A stochastic LTAG derivation proceeds as follows (Schabes 1992, Resnik 1992). An initial tree is selected with probability P_i and subsequent substitutions are performed with probability P_s and adjunctions are performed with probability P_a.

For each τ that can be valid start of a derivation:

$$\sum_{\tau} P_i(\tau) = 1$$

Each subsequent substitution or adjunction occurs independently. For possible substitutions defined by the grammar:

$$\sum_{\tau'} P_s(\tau, \eta \rightarrow \tau') = 1$$

where, τ' is substituting into node η in tree τ. For possible adjunctions in the grammar there is an additional factor which is required for the probability to be well-formed:

$$P_a(\tau, \eta \rightarrow \textsc{na}) + \sum_{\tau'} P_a(\tau, \eta \rightarrow \tau') = 1$$

$P_a(\tau, \eta \rightarrow \textsc{na})$ is the probability that there is no adjunction (NA) at node η in τ.

Each LTAG derivation $\mathcal{D}$ is built starting from some initial tree α. Let us consider the probability of a derivation $\mathcal{D}$ which was constructed using p substitutions and q adjunctions and r internal nodes which had no adjunction. If we assume, for simplicity that each elementary tree is lexicalized by exactly one word, then the length of the sentence $n = p + q + 1$.

(14.37)
$$\begin{aligned} &Pr(\mathcal{D}, w_0 \ldots w_n) = \\ &\quad P_i(\alpha, w_i) \times \prod_p P_s(\tau, \eta, w \rightarrow \tau', w') \times \\ &\quad \prod_q P_a(\tau, \eta, w \rightarrow \tau', w') \times \\ &\quad \prod_r P_a(\tau, \eta, w \rightarrow \textsc{na}) \end{aligned}$$

This derivation $\mathcal{D}$ can be drawn graphically as a tree where each node in this derivation tree is an elementary tree in the original LTAG.

P_i and P_s can be written as the following conditional probabilities which can be estimated from the training data. For further details about decomposing these probabilities further to make parsing and decoding easier and details about prior probabilities see chapter 16 in this volume.

$$\begin{aligned} P_i(\alpha, w \mid TOP) \\ P_s(\alpha, w' \mid \tau, \eta, w) \\ P_a(\beta, w' \mid \tau, \eta, w) \end{aligned}$$

The probability of a sentence S computed using this model is the sum of all the possible derivations of the sentence.

$$P(S) = \sum_D Pr(D, S)$$

A generative model can be defined instead of a conditional probability to obtain the best derivation $\mathcal{D}_{\text{BEST}}$ given a sentence S. The value for (14.38) is computed using the Equation 14.37.

$$\begin{aligned}\mathcal{D}_{\mathrm{BEST}} &= \arg\max_{\mathcal{D}} \Pr(\mathcal{D} \mid S) \\ &= \arg\max_{\mathcal{D}} \frac{\Pr(\mathcal{D}, S)}{\Pr(S)} \\ &= \arg\max_{\mathcal{D}} \Pr(\mathcal{D}, S) \qquad (14.38)\end{aligned}$$

The particular definition of a stochastic TAG is by no means the only way of defining a probabilistic grammar formalism with TAG. There have been some variants from the standard model that have been published since the original stochastic TAG papers.

For example, the restriction of one adjunction per node could be dropped and a new variant of standard TAG can be defined which permits arbitrary number of modifications per node. This variant was first introduced by (Schabes and Shieber 1992, Schabes and Shieber 1994). Tree Insertion Grammar (Schabes and Waters 1995) is a variant of TAG where the adjoining operation is restricted in a certain way and this restricted operation is named insertion. TIGs are weakly equivalent to CFGs but they can produce structural descriptions that are not obtainable by any CFG.

A stochastic version of insertion (Schabes and Waters 1996) was defined in the context of Tree Insertion Grammar. In this model, multiple trees can be adjoined to the left and to the right of each node with the following probabilities:

$$P_{la}(\tau, \eta \rightarrow \mathrm{NA}_l) + \sum_{\tau'} P_{la}(\tau, \eta \rightarrow \tau') = 1$$
$$P_{ra}(\tau, \eta \rightarrow \mathrm{NA}_r) + \sum_{\tau'} P_{ra}(\tau, \eta \rightarrow \tau') = 1$$

There are many other probability measures that can be used with TAG and its variants. One can easily go beyond the bi-lexical probabilities that have been the main focus in this paper to probabilities that invoke greater amounts of structural or lexical context. (Carroll and Weir 1997), for example, gives some additional probability models one might consider useful when using TAGs.

14.5 Consequences of applying probability measures to Tree Adjoining Languages

In this section we look at some formal properties of stochastic TAGs. To gain some intuition about probability assignments to tree adjoining languages, let us take for example, a language well known to be a tree adjoining language:

$$L(G) = \{a^n b^n c^n d^n | n \geqslant 1\}$$

It seems that we should be able to use a function ψ to assign any probability distribution to the strings in $L(G)$ and then expect that we can assign appropriate probabilites to the adjunctions in G such that the language generated by G has the same distribution as that given by ψ. However a function ψ that grows smaller by repeated multiplication as the inverse of an exponential function cannot be matched by any TAG because of the *constant growth* property of TAGs (see (Vijay-Shanker 1987), p. 104). An example of such a function ψ is a simple Poisson distribution (14.39), which in fact was also used as the counterexample in (Booth and Thompson 1973) for CFGs, since CFGs also have the constant growth property.

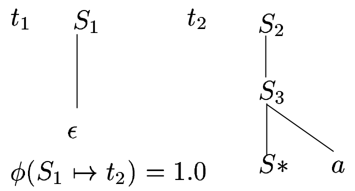
(14.39) $$\psi(a^n b^n c^n d^n) = \frac{1}{e \cdot n!}$$

This shows that probabilistic TAGs, like CFGs, are constrained in the probabilistic languages that they can recognize or learn. As shown above, a probabilistic language can fail to have a generating probabilistic TAG.

The reverse is also true: some probabilistic TAGs, like some CFGs, fail to have a corresponding probabilistic language, i.e. they are not consistent. There are two reasons why a probabilistic TAG could be inconsistent: "dirty" grammars, and destructive or incorrect probability assignments.

"Dirty" grammars. Usually, when applied to language, TAGs are lexicalized and so probabilities assigned to trees are used only when the words anchoring the trees are used in a derivation. However, if the TAG allows non-lexicalized trees, or more precisely, auxiliary trees with no yield, then looping adjunctions which never generate a string are possible. However, this can be detected and corrected by a simple search over the grammar. Even in lexicalized grammars, there could be some auxiliary trees that are assigned some probability mass but which can never adjoin into another tree. Such auxiliary trees are termed *unreachable* and techniques similar to the ones used in detecting unreachable productions in CFGs can be used here to detect and eliminate such trees.

Destructive probability assignments. This problem is a more serious one. Consider the probabilistic TAG shown in (14.40)[74].



(14.40)

$$\phi(S_1 \mapsto t_2) = 1.0$$
$$\phi(S_2 \mapsto t_2) = 0.99$$
$$\phi(S_2 \mapsto nil) = 0.01$$
$$\phi(S_3 \mapsto t_2) = 0.98$$
$$\phi(S_3 \mapsto nil) = 0.02$$

Consider a derivation in this TAG as a generative process. It proceeds as follows: node S_1 in t_1 is rewritten as t_2 with probability 1.0. Node S_2 in t_2 is 99 times more likely than not to be rewritten as t_2 itself, and similarly node S_3 is 49 times more likely than not to be rewritten as t_2. This however, creates two more instances of S_2 and S_3 with same probabilities. This continues, creating multiple instances of t_2 at each level of the derivation process with each instance of t_2 creating two more instances of itself. The grammar itself is not malicious; the probability assignments are to be blamed. It is important to note that inconsistency is a problem even though for any given string there are only a finite number of derivations, all halting. Consider the probability mass function (*pmf*) over the set of all derivations for this grammar. An inconsistent grammar would have a *pmf* which assigns a large portion of probability mass to derivations that are non-terminating. This means there is a finite probability the generative process can enter a generation sequence which has a finite probability of non-termination.

14.5.1 Conditions for Consistency

A probabilistic TAG G is *consistent* if and only if:

(14.41)
$$\sum_{v \in L(G)} \Pr(v) = 1$$

where $\Pr(v)$ is the probability assigned to a string in the language. If a grammar G does not satisfy this condition, G is said to be inconsistent. Note that this is a very general definition and consistency can be defined for all well-defined generative processes (Harris 1963).

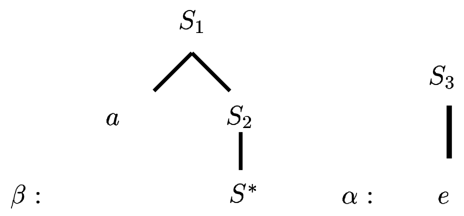
To show that a given probabilistic TAG is consistent we can exploit context-free nature of TAG derivations and exploit the existing result

[74]The subscripts are used as a simple notation to uniquely refer to the nodes in each elementary tree. They are not part of the node label for purposes of adjunction.

by (Booth and Thompson 1973, Wetherell 1980) which provides conditions under which stochastic CFGs can be shown to be consistent[75].

We explain the alternative method using an example. Consider a PTAG $G = \langle\{\beta, \alpha\}, \phi\rangle$ with trees defined in (14.42) and parameter values given in (14.43).

(14.42)



(14.43)

$$\begin{aligned}
\phi(S_1 \rightarrow \beta) &= 0.99 \\
\phi(S_1 \rightarrow \epsilon) &= 0.01 \\
\phi(S_2 \rightarrow \beta) &= 0.98 \\
\phi(S_2 \rightarrow \epsilon) &= 0.02 \\
\phi(S_3 \rightarrow \beta) &= 1.0 \\
\phi(S_3 \rightarrow \epsilon) &= 0.0
\end{aligned}$$

G is an inconsistent PTAG (it assigns probability mass to derivations that do not terminate). *Can we detect this inconsistency by using the Booth and Thompson result on some PCFG G' constructed by examining the PTAG G ?*

The Booth and Thompson result can be stated as follows:

Theorem 3 *A probabilistic grammar is consistent if the* spectral radius $\rho(\mathcal{M}) < 1$*, where $\mathcal{M}$ is the stochastic expectation matrix computed from the context-free grammar. (Booth and Thompson 1973, Soule 1974)*

This theorem provides a way to determine whether a grammar is consistent. All we need to do is compute the spectral radius of the expectation matrix $\mathcal{M}$, which defines the expected number of times each CFG rule will be seen in a corpus of parse trees generated by the stochastic CFG. The spectral radius is equal to the modulus of the largest eigenvalue of $\mathcal{M}$. If this value is less than one then the grammar is consistent.

In (14.44) we show a set of productions (rules) P constructed from the PTAG G designed to show the derivations possible in G.

[75]Thanks to Steve Abney for discussions on this topic.

(14.44)
$$\begin{array}{rcl} \alpha & \rightarrow & S_3 \\ S_3 & \rightarrow & \beta \mid \epsilon \\ \beta & \rightarrow & S_1\, S_2 \\ S_1 & \rightarrow & \beta \mid \epsilon \\ S_2 & \rightarrow & \beta \mid \epsilon \end{array}$$

To use the Booth and Thompson result we need to satisfy two conditions. The first condition is that the PCFG is *proper*. That is the parameters of the PCFG satisfy Eqn. 14.45.

(14.45)
$$\sum_{(X \rightarrow Y) \in P} \mathcal{P}(X \rightarrow Y \mid X) = 1$$

The rule probabilities for the CFG in (14.44) derived from the PTAG parameters in (14.43) are shown in (14.46).

(14.46)
$$\begin{array}{rcl} \mathcal{P}(\alpha \rightarrow S_3 \mid \alpha) & = & \sum_X \mathcal{P}(S_3 \rightarrow X \mid S_3) = 1.0 \\ \mathcal{P}(S_3 \rightarrow \beta \mid S_3) & = & 1.0 \\ \mathcal{P}(S_3 \rightarrow \epsilon \mid S_3) & = & 0.0 \\ \mathcal{P}(\beta \rightarrow S_1\, S_2 \mid \beta) & = & \sum_X \mathcal{P}(S_1 \rightarrow X \mid S_1) \times \sum_Y \mathcal{P}(S_2 \rightarrow Y \mid S_2) = 1.0 \\ \mathcal{P}(S_1 \rightarrow \beta \mid S_1) & = & 0.99 \\ \mathcal{P}(S_1 \rightarrow \epsilon \mid S_1) & = & 0.01 \\ \mathcal{P}(S_2 \rightarrow \beta \mid S_2) & = & 0.98 \\ \mathcal{P}(S_2 \rightarrow \epsilon \mid S_2) & = & 0.02 \end{array}$$

Based on these probabilities we can compute the expectation matrix $\mathcal{M}$ using the method defined in (Booth and Thompson 1973).

$$\mathcal{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.9900 \\ 0 & 0 & 0 & 0 & 0.9800 \\ 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 1.0000 & 0 & 0 \\ 1.0000 & 1.0000 & 0 & 0 & 0 \end{bmatrix}$$

The eigenvalues of the expectation matrix come out to be $0, 1.4036, -1.4036$. Since the largest eigenvalue is greater than 1, we correctly predict that the original PTAG G is inconsistent.

A formal justification for this method of showing whether a given stochastic TAG is consistent is given in (Sarkar 1998) by showing a reduction of the TAG derivation process to a multitype Galton-Watson branching process (Harris 1963).

14.6 Related Work

In this section we give an overview of some additional work done within the area of stochastic Tree Adjoining Grammars. We also compare the work in stochastic TAGs with work done in neighbouring areas like Data Oriented Parsing.

14.6.1 Work in Stochastic TAG and Related Areas

(Hwa 1998) uses the inside-outside algorithm for TAGs given in (Schabes 1992) and applies it to stochastic Tree Insertion Grammars. The algorithm is combined with the use of bracketed Treebank data (Pereira and Schabes 1992) as a source of a partial bracketing of the training sentences. The experiments reported were conducted on the WSJ Penn Treebank with the input to the learning algorithm being part-of-speech tags (rather than the words themselves). (Hwa 1999) extends the partial bracketing approach by suppressing various kinds of labeled brackets as a possible way of minimizing annotation cost by recovering some labeled brackets automatically.

Chiang (2000) (also chapter 16, this volume) gives a statistical parser based on stochastic Tree Insertion Grammars. The experiments were based on fully lexicalized elementary trees and achieves 87.6% labeled precision and 87.4% labeled recall. These results show that one does not have to sacrifice performance over lexicalized PCFGs while maintaining a more elaborate model using TAGs. Chiang (chapter 16, this volume) also reports results on the Chinese Treebank. This involved only minor changes to the English parser.

(Xia et al. 2001, Xia 2001) reports on algorithms that permit the extraction of TAG derivation trees from Treebanks in various languages. The algorithms use only minimal edits to tables of data that are localized to each new Treebank.

(Nederhof et al. 1998) shows that it is possible to extend the notion of assigning a probability to an entire sentence (defined in Section 14.4) to an arbitrary prefix of an input sentence. This extends a result shown for stochastic CFGs by (Jelinek and Lafferty 1991, Stolcke 1995).

(Sarkar 2001) explores some new machine learning techniques to enable statistical parsers to take advantage of unlabeled data. By exploiting the representation of stochastic TAG to view parsing as a classification task, it uses a machine learning method called Co-Training to iteratively label new training data for the parser, improving its performance over simply using the available amount of labeled data. While training only on the labeled set gave a performance of 72.23% and 69.1% labeled bracketing precision and recalll, the technique achieves 80.02% and 79.64% labeled bracketing precision and recall using Co-Training with a labeled set of about 10K sentences and an unlabeled set of 30K sentences. While these results concentrate on training from a small labeled set, current experiments extending these results are focused on improving performance when using larger amounts (upto a million words) of labeled data using upto 23 million words of unlabeled data.

(Srinivas and Joshi 1999, Srinivas 1997b, Srinivas 1997a) describes a method of partial parsing that uses local attachment heuristics after a probabilistic method that picks the best elementary tree for each word in a sentence: a technique termed as SuperTagging indicating the affinity between the problems of assigning complex structures such as trees to each word in a sentence as compared to the assignment of part of speech tags.

More distantly related use of elementary trees (of depth greater than 1) for statistical parsing occur in the works of (Sima'an 2000) and (Skut and Brants 1998). The work reported in (Skut and Brants 1998) is related to the use of SuperTagging for partial parsing in the work of (Srinivas 1997a).

14.6.2 Comparison with Other Work

Chiang, (chapter 16, this volume), points out several similarities and differences between statistical parsing using TAGs and other tree-based statistical parsers like DOP (Bod 2001). We briefly summarize some of these points here. The DOP1 model is related to stochastic TAGs since they both use probability measures over tree grammars. However, a stochastic TAG parser computes the best derivation tree for the input sentence, while DOP1 computes the best derived tree (or parse tree) by summing over all possible derivations that yield that tree. Here we have argued for the primacy of the derivation tree over the parse tree. Computing the best derivation provides an interface to future processing, for example, to compute a semantics for the input sentence while according to the TAG viewpoint, the derived tree provides no such benefit. Other benefits of using the DOP approach are the use of non-lexicalized trees and the use of trees with multiple lexical items at its leaf nodes. The use of both of these kinds of trees are already possible in the non-statistical uses of TAG (such as the parser used to parse the XTAG English Grammar (XTAG Group 2001)). Extensions to stochastic TAGs and their use in statistical parsing have not yet shown improvements over the simpler form of stochastic TAG which is purely lexicalized and has only one lexical item per elementary tree.

Hoogweg (chapter 17, this volume) considers an extension of DOP with the operation of insertion taken from Tree Insertion Grammar. Hoogweg gives a comparative analysis of his experiments with DOP with and without the use of insertion. Hoogweg maintains the primacy of the parse tree where all possible derivations of a parse tree are computed, while in stochastic TAG and related work only the best derivation tree is ever computed based on a generative probability model. In linguistic terms, in an LTAG the elementary trees are semantically encapsulated objects and the derivation tree in LTAG leads to a compositional semantics. This is not a necessary requirement for DOP.

14.7 Conclusion

We have given an introduction to a formalism called Tree Adjoining Grammars (TAGs) that is useful in defining linguistic descriptions that are structurally complex. TAGs accomplish this by using trees, or even directed acyclic graphs, as elementary objects. Also in this paper we show how these descriptions are useful in the context of statistical parsing. We provide a review of the definitions and results in the field of stochastic TAGs and show how the definition of stochastic TAG is well-defined. Finally, we have provided a brief comparison of stochastic TAGs and related work as well other tree based work in parsing such as DOP.

References

Bod, Rens. 2001. What is the Minimal Set of Fragments that Achieves Maximal Parse Accuracy. In *Proceedings of ACL-2001.* Toulouse, France.

Booth, T. L., and R. A. Thompson. 1973. Applying Probability Measures to Abstract Languages. *IEEE Transactions on Computers* C-22(5):442–450.

Carroll, J., and D. Weir. 1997. Encoding Frequency Information in Lexicalized Grammars. In *Proc. 5th Int'l Workshop on Parsing Technologies IWPT-97.* Cambridge, Mass.

Chen, John, and K. Vijay-Shanker. 2000. Automated Extraction of TAGs from the Penn Treebank. In *Proc. of the 6th International Workshop on Parsing Technologies (IWPT-2000), Italy.*

Chiang, David. 2000. Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *Proc. of ACL-2000.*

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

Harris, T. E. 1963. *The Theory of Branching Processes.* Berlin: Springer-Verlag.

Hwa, R. 1998. An Empirical Evaluation of Probabilistic Lexicalized Tree Insertion Grammars. In *Proc. of COLING-ACL '98*, 557–563. Montreal, Canada.

Hwa, R. 1999. Supervised Grammar Induction using Training Data with Limited Constituent Information. In *Proceedings of the 37th Annual Meeting of the ACL*, 73–79. June.

Jelinek, F., and J. Lafferty. 1991. Computation of the Probability of Initial Substring Generation by Stochastic Context-Free Grammars. *Computational Linguistics* 17(3):315–323.

Joshi, Aravind. 1990. Processing crossed and nested dependencies: An automaton perspective on the psycholinguistic results. *Language and Cognitive Processes* 5(1):1–27.

Joshi, Aravind, and Yves Schabes. 1997. Tree Adjoining Grammars. In *Handbook of Formal Languages and Automata*, ed. G. Rozenberg and A. Salomaa. Springer-Verlag.

Joshi, Aravind, and K. Vijay-Shanker. 1999. Compositional Semantics with LTAG: How Much Underspecification Is Necessary? In *Proc. of 3nd International Workshop on Computational Semantics.*

Joshi, Aravind K. 1985. Tree Adjoining Grammars: How much context Sensitivity is required to provide a reasonable structural description. In *Natural Language Parsing*, ed. D. Dowty, I. Karttunen, and A. Zwicky. 206–250. Cambridge, U.K.: Cambridge University Press.

Lari, K., and S. J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language* 4:35–56.

Nederhof, Mark-Jan, Anoop Sarkar, and Giorgio Satta. 1998. Prefix Probabilities from Probabilistic Tree Adjoining Grammars. In *Proceedings of COLING-ACL 1998.* Montreal.

Pereira, Fernando, and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics.*

Rambow, O., and A. Joshi. 1995. A Formal Look at Dependency Grammars and Phrase-Structure Grammars, with Special Consideration of Word-Order Phenomena. In *Current Issues in Meaning-Text Theory*, ed. Leo Wanner. London: Pinter.

Resnik, Philip. 1992. Probabilistic Tree-Adjoining Grammars as a framework for statistical natural language processing. In *Proc. of COLING '92*, 418–424. Nantes, France.

Sarkar, A. 1998. Conditions on Consistency of Probabilistic Tree Adjoining Grammars. In *Proceedings of COLING-ACL '98*, 1164–1170. Montreal.

Sarkar, Anoop. 2001. Applying Co-Training Methods to Statistical Parsing. In *Proceedings of NAACL 2001.* Pittsburgh, PA, June.

Schabes, Y. 1992. Stochastic Lexicalized Tree-Adjoining Grammars. In *Proc. of COLING '92*, 426–432. Nantes, France.

Schabes, Y., and S. Shieber. 1992. An Alternative Conception of Tree-Adjoining Derivation. In *Proceedings of the* 20^{th} *Meeting of the Association for Computational Linguistics.*

Schabes, Y., and R. Waters. 1996. Stochastic Lexcalized Tree-Insertion Grammar. In *Recent Advances in Parsing Technology*, ed. H. Bunt and M. Tomita. 281–294. Kluwer.

Schabes, Yves, and Stuart Shieber. 1994. An Alternative Conception of Tree-Adjoining Derivation. *Computational Linguistics* 20(1):91–124.

Schabes, Yves, and Richard Waters. 1995. Tree Insertion Grammar: A Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced. *Computational Linguistics* 21(4):479–513.

Sima'an, K. 2000. Tree-gram Parsing: Lexical Dependencies and Structural Relations. In *Proceedings of the 38th Annual Meeting of the ACL*, 53–60. Hong Kong.

Skut, Wojciech, and Thorsten Brants. 1998. A Maximum-Entropy Partial Parser for Unrestricted Text. In *Proceedings of the Sixth Workshop on Very Large Corpora.* Montreal, Canada.

Soule, S. 1974. Entropies of probabilistic grammars. *Inf. Control* 25:55–74.

Srinivas, B. 1997a. *Complexity of Lexical Descriptions: Relevance to Partial Parsing.* Doctoral dissertation, University of Pennsylvania.

Srinivas, B. 1997b. Performance Evaluation of SuperTagging for Partial Parsing. In *Fifth International Workshop on Parsing Technologies.* Boston, September.

Srinivas, B., and Aravind Joshi. 1999. Supertagging: An approach to Almost Parsing. *Computational Linguistics* 25(2).

Stolcke, A. 1995. An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities. *Computational Linguistics* 21(2):165–201.

Vijay-Shanker, K. 1987. *A Study of Tree Adjoining Grammars.* Doctoral dissertation, Department of Computer and Information Science, University of Pennsylvania.

Wetherell, C. S. 1980. Probabilistic Languages: A Review and Some Open Questions. *Computing Surveys* 12(4):361–379.

Xia, Fei. 2001. *Investigating the Relationship between Grammars and Treebanks for Natural languages.* Doctoral dissertation, University of Pennsylvania, Philadelphia, PA.

Xia, Fei, Chunghye Han, Martha Palmer, and Aravind Joshi. 2001. Automatically Extracting and Comparing Lexicalized Grammars for Different Languages. In *Proc. of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001).* Seattle, Washington.

XTAG Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03. IRCS, University of Pennsylvania.