Supertagging: Using Complex Lexical Descriptions in Natural Language Processing

Edited by Srinivas Bangalore and Aravind K. Joshi

A Bradford Book
The MIT Press
Cambridge, Massachusetts
London, England

This book was set in LaTex by the authors.

Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Supertagging : using complex lexical descriptions in natural language processing / edited by Srinivas Bangalore and Aravind K. Joshi.
p. cm. — (A Bradford Book)
Includes bibliographical references and index.
ISBN 978-0-262-01387-1 (hardcover : alk. paper) 1. Natural language processing (Computer science) 2. Computational linguistics—Statistical methods. 3. Semantics. 4. Lexicology. I.
Bangalore, Srinivas, 1969– II. Joshi, Aravind K. (Aravind Krishna), 1929–
QA76.9.N38S86 2010
006.3'5—dc22
2009028194

10 9 8 7 6 5 4 3 2 1

1

Introduction

Srinivas Bangalore and Aravind Joshi

The conventional (mathematical) wisdom in specifying a grammar formalism is to start with basic primitive structures as simple as possible and then introduce various operations for constructing more complex structures. These operations can be simple or complex and the number of operations (although finite) need not be limited. New operations (simple or complex) can be introduced in order to describe more complex structures.

An alternate approach is to start with complex (more complicated) primitives, which capture directly some crucial linguistic properties and then introduce some general operations for composing these complex structures (primitive or derived). What is the nature of these complex primitives? In the conventional approach the primitive structures (or rules) are kept as simple as possible. This has the consequence that information (e.g., syntactic and semantic) about a lexical item (word) is distributed over more than one primitive structure. Therefore the information associated with a lexical item is not captured locally, that is, within the domain of a primitive structure. We will illustrate this in section 1.1 in terms of the well-known context-free grammar (CFG) framework.

In contrast, in the alternate approach described in this book, we allow the primitive structures to be as complex as necessary to capture all the relevant information about a lexical item in the local domain of a primitive structure. We refer to the primitive structure as a *supertag* and is associated with a lexical item. A supertag needs to localize the following information: (a) a lexical item taken as a predicate has zero or more arguments, (b) the arguments need to satisfy certain syntactic and semantic constraints, which are determined by the lexical item, and (c) the different positions that the arguments will occupy relative to the position of the lexical item. Hence, in the alternate approach, all the pieces of information associated with a lexical item have to be represented in the local domains of the primitive structures of the formal system. Although the primitives are complex and there may be more than one primitive structure associated with a lexical item, the number of primitives is finite. Further the combining operations are kept to the minimum and they are language independent (that is, *universal*). In this approach nonlocal dependencies are pushed to become local, that is, these dependencies start out in the basic primitive structure (supertag) and hence we characterize this approach as *complicate locally, simplify globally* (CLSG). The architecture resulting from the CLSG approach has important implications for linguistics, computational linguistics, and psycholinguistics, including generation and acquisition.

There is another dimension in which formal systems can be characterized. One could start with an unconstrained formal system (Turing machine equivalent, for example) and then add linguistic constraints, which become in a sense, all stipulative. Alternatively, one could start with a formal system that is constrained and just adequate for describing language. The formal constraints then become universal, in a sense. All other linguistic constraints become stipulative and language specific. Now it turns out that the CLSG approach leads to constrained formal systems. This convergence is of interest in its own right which is not discussed in this book. Our focus will be on the CLSG approach and its implications for the architecture of the grammars, their processors, and how it has been realized in different grammar formalisms.

We begin this book by illustrating these ideas in terms of the Lexicalized Tree-Adjoining grammar (LTAG), a class of grammars that illustrates the CLSG approach by adopting it in its extreme form in section 1.3. LTAG and some of its extensions have been investigated both formally and computationally for over twenty five years (See Joshi et al., 1975; Joshi, 1985; Kroch and Joshi, 1985; Vijay-Shanker, 1987; Weir, 1988; Kroch, 1989; Kroch and Santorini, 1991; Schabes, 1992; Rambow, 1994; Resnik, 1992; Chiang, 2000; Sarkar, 2002; Abeillé, 2002; Prolo, 2003). There are several formal systems that are clearly related to LTAG. Some examples are Combinatory Categorial Grammars (CCG) (Steedman, 1996), Stabler's version of minimalist grammars (Stabler, 1997), Lexical Functional Grammars (LFG) (Kaplan and Bresnan, 1983), Head Driven Phrase Structure Grammars (HPSG) (Pollard and Sag, 1994) (for a constrained version of HPSG, (see Kasper et al., 1995). Linear Indexed Grammars (LIG) by Gazdar, and Head Grammars (HG) by Pollard. CCG and LTAG have been shown to be weakly equivalent, that is, in terms of the string sets they generate but not in terms of the structural descriptions. These relationships have been discussed extensively in Joshi et al. (1991).

In this chapter, in section 1.1, we will introduce the notions of domain of locality and lexicalization in the context of the well-known context-free grammars (CFG) and then show how lexicalized tree-adjoining grammars (LTAG) arise in the process of lexicalizing CFGs and extending the domain of locality in section 1.2. We will also show how the architecture of the building blocks of LTAG directly predicts many complex dependency patterns and then summarize some important properties of LTAG in section 1.3. In section 1.4, we introduce the perspective of supertagging for LTAG and discuss its implications for language description and language processing. In section 1.5, we mention the relevance of supertagging for psycholinguistic models of sentence processing. In section 1.6, we group the chapters of this book under thematic topics and briefly summarize their contributions towards the goal of this book.

1.1 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 1.1). 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 $S \rightarrow NP\ V\ NP$. 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 1.1) is lexicalized.

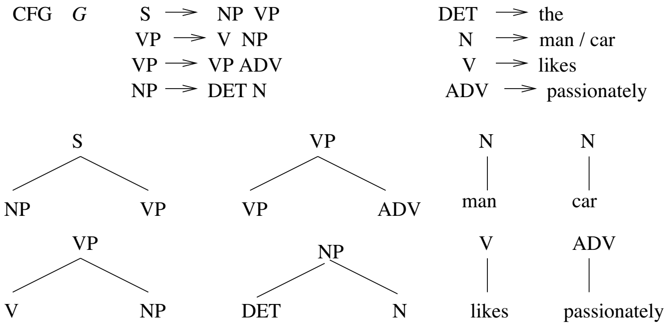


FIGURE 1.1 Domain of locality of a context-free grammar

The five rules on the right are lexicalized; that is, they have a lexical anchor. The rules on the left are not lexicalized. The second, the third, and the fourth rule on the left are almost lexicalized, in the sense that they each have at least one preterminal category (V in the second rule, ADV in the third rule, and DET and N in the fourth rule); that is, by replacing V by *likes*, ADV by *passionately*, and either DET by *the* or N by *man*, these three rules will become lexicalized. However, the first rule on the left ($S \rightarrow NP\ VP$) cannot be lexicalized, not certainly by *man*.

Can a CFG be lexicalized that is, 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 (that is, the tree language of G) is the same as the tree language $T(G')$ of G'? Of course, if we require that only the string languages of G and G' be the same (that is, 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 (see Linz, 2001) 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.[1] We call this *weak* lexicalization. The lexicalization we are interested in requires the tree languages (that is, the set of structural descriptions) to be the same (that is, strong equivalence). We call this *strong* lexicalization. It is easily seen, even from the example in figure. 1.1, that a nonlexicalized CFG cannot be necessarily strongly lexicalized by another CFG. Basically this follows from the fact that the domain of locality of CFG is a one level tree corresponding to a rule in the grammar (for detail, see Joshi and Schabes, 1997). In section 1.2, we will consider lexicalization of CFG by larger (extended) domain of locality.

Before proceeding, it would be helpful to review certain definitions. The primitive structures of a formalism (also called elementary structures or elementary trees, as special cases) provide a *local* domain for specifying linguistic constraints (pieces of linguistic theory) in the sense that if the constraints are specifiable by referring to just the structures that are associated with the elementary structures then it is specifiable over the domain of these elementary structures. Therefore, we refer to the domains corresponding to the elementary structures as *domains of locality*. Formalism A is said to provide an *extended domain of locality* as compared to a formalism B if there is a linguistic constraint that is not specifiable in the local domains associated with B but which is specifiable in the local domains associated with A. The goal of the CLSG approach is to look for a formalism that provides local domains large enough so that, in principle, *all* linguistic constraints (pieces of linguistic theory) can be specified over these local domains. In the conventional approach (e.g., CFG-based) the specification of a constraint is often spread out over more than one local domain, and thus the specification of a constraint is intertwined with how the local domains are composed by the grammar; in other words, specification of a constraint will require specification of recursion, resulting in an effectively unbounded domain. In contrast, in the CLSG approach we seek a system with extended (but still finite) domains of locality capable of specifying the linguistic constraints over these extended domains. Thus, recursion does not enter into the specification of the constraints. We call this property as *factoring recursion away from the domains of locality.*

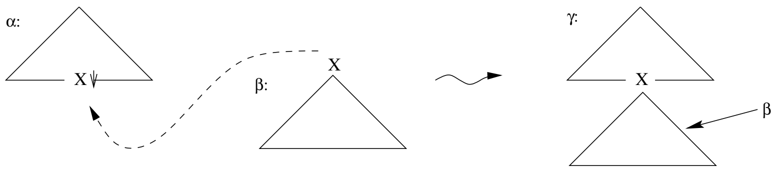


FIGURE 1.2 Substitution

1.2 Lexicalization of CFGs by Grammars with Larger Domains of Locality

Now we can ask the following question. Can we strongly lexicalize a CFG by a grammar with a larger domain of locality? Figure 1.2 and figure 1.3 show a tree substitution grammar where the elementary objects (building blocks) are the three trees in figure 1.3 and the combining operation is the *tree substitution* operation

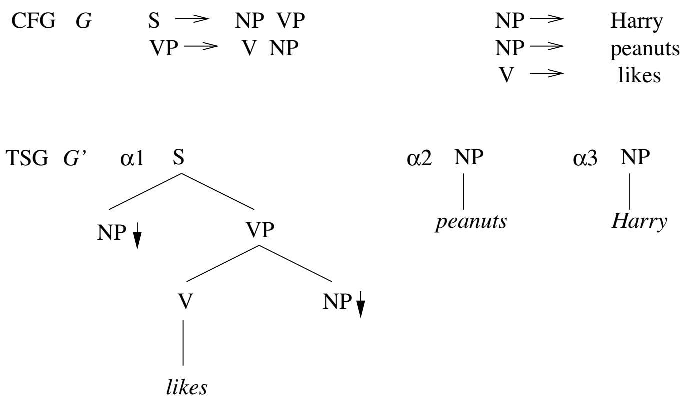


FIGURE 1.3 Tree substitution grammar

shown in figure 1.2. Note that each tree in the tree substitution grammar (TSG), G' is lexicalized; that is, it has a *lexical anchor*. It is easily seen that G' indeed strongly lexicalizes G. However, TSGs fail to strongly lexicalize CFGs in general. We show this by an example. Consider the CFG, G, in figure 1.4 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. In order for a tree to grow on both sides, the distance between the lexical anchor of a tree, a, and the root of the tree, S, must become arbitrarily large. Substitution makes a tree grow only at the leaves of the tree and cannot make it grow internally. Hence, the TSG, G', cannot strongly lexicalize the CFG, G (Joshi and Schabes, 1997). Thus, even with the extended domain of locality of TSGs, we cannot strongly lexicalize CFGs as long as substitution is the only operation for putting trees together.

We now introduce a new operation called *adjoining*, as shown in figure 1.5. Adjoining involves splicing (inserting) one tree into another. More specifically, a tree β 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 (terminal or nonterminal) on the frontier of β, the nonterminal nodes 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

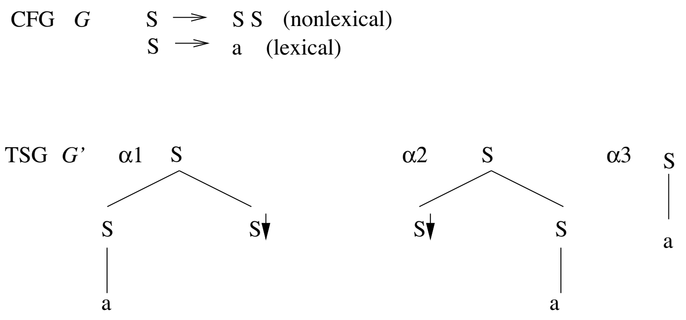


FIGURE 1.4 A tree substitution grammar

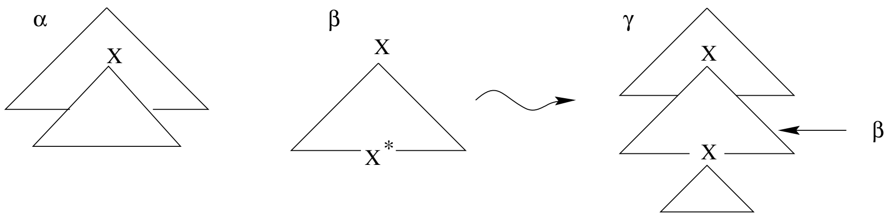
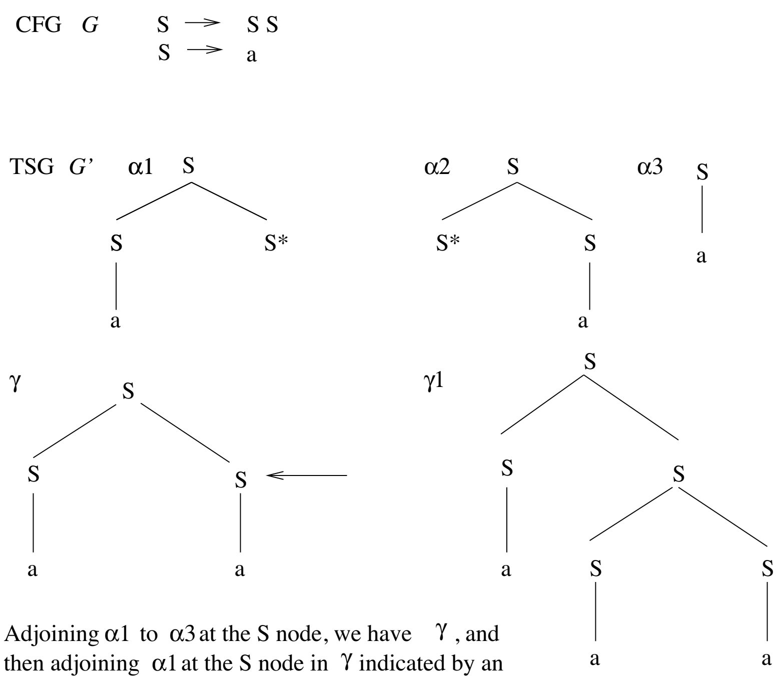


FIGURE 1.5 Adjoining



Adjoining α1 to α3 at the S node, we have γ, and then adjoining α1 at the S node in γ indicated by an arrow, we have γ1.

FIGURE 1.6 Adjoining arises out of lexicalization

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 tree*, 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 1.6, 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 1.4. However, now trees α_1 and α_2 are auxiliary trees (foot node 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.

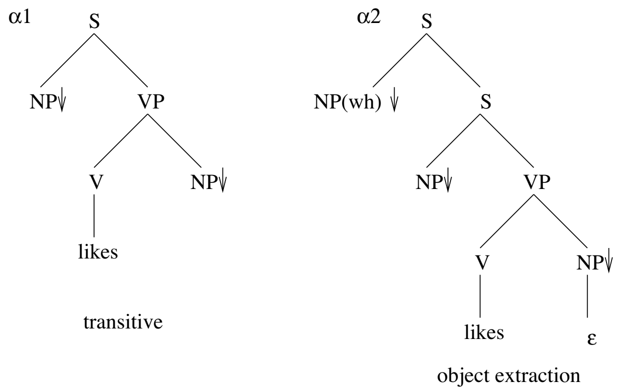


FIGURE 1.7 LTAG: Elementary trees for *likes*

1.3 Lexicalized Tree-Adjoining Grammars

Rather than give formal definitions for LTAG and derivations in LTAG, we will give a simple example to illustrate some key aspects of LTAG.[2] We show some elementary trees of a toy LTAG grammar for English. Figure 1.7 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 α_2 the node for $NP(wh)$ (the extracted argument) has to be in the tree for *likes*. Further, there is a dependency between the $NP(wh)$ node and the NP node, which is the complement of *likes* (that is, to the right of V dominating *likes*), and this dependency is local to α_2. The tree α_2 shows not only that $NP(wh)$ is an argument of *likes* but also that it is large enough to indicate a specific structural position for that argument.

Therefore, 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 that 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. This will become clear as soon as we describe the derivation in figure 1.8.

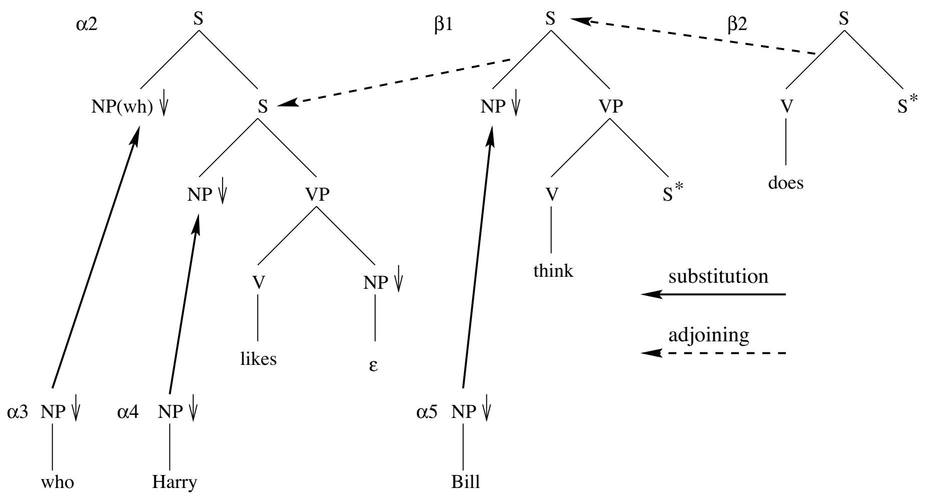


FIGURE 1.8 LTAG derivation for *who does Bill think Harry likes*

Figure 1.9 shows some additional elementary trees; trees α_3, α_4, and α_5 and trees β_1 and β_2. The β trees with foot nodes marked with * will enter a derivation by the operation of adjoining. The α trees enter a derivation by the operation of substitution.[3]

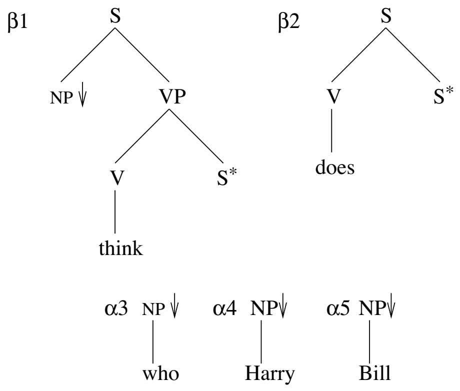


FIGURE 1.9 LTAG: Sample elementary trees

A derivation using the trees α_2, α_3, α_4, α_5, β_1, and β_2 is shown in figure 1.8. The trees for *who* and *Harry* are substituted in the tree for *likes* at the respective NP nodes, at node addresses 1 and 2.1 in α_2. The tree for *Bill* is substituted in the tree for *think* at the NP node at the node address 1 in β_1. The tree for *does* is adjoined to the root node (address 0) 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 at the address 2 in α_2. This derivation results in the *derived tree* for

Who does Bill think Harry likes

as shown in figure 1.10. 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 1.10. It has become *long distance*. However, it started out as a local dependency. A key property of LTAGs is that all dependencies are local, that is, they are specified in the elementary trees. They can become long distance as a result of the composition operations. Figure 1.10 is the conventional tree associated with the sentence.

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 1.11. The nodes of the tree are labeled by the tree labels such as α_2 together with its lexical anchor *likes*[4]. The number on an edge of a derivation tree refers to the node address in a tree into which either a substitution or adjoining has been made. Thus, for example, in figure 1.11 the $\alpha_3(who)$ tree is substituted at the node with address 1 in the tree $\alpha_2(likes)$, the tree $\beta_1(thinks)$ is adjoined at the address 2 in the tree $\alpha_2(likes)$, and so on. Solid edges denote substitution, and dotted edges denote adjoining.

The derivation tree is the crucial derivation structure for LTAG. It records the history of composition in terms of the elementary trees (primitive building blocks)

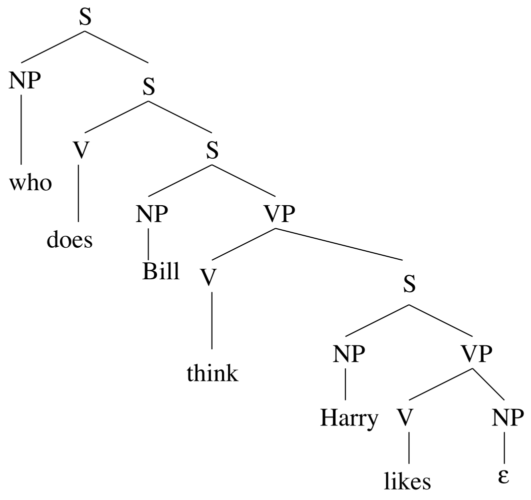


FIGURE 1.10 LTAG derived tree for *who does Bill think Harry likes*

of LTAG. The derived tree in figure 1.10 does not indicate what the component elementary trees are for the final derived tree. It should be clear that from the derivation tree we can always obtain the derived tree by performing the substitutions and adjoinings indicated in the derivation tree. So in this sense the derived tree is redundant.

Further, 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 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; Kallmeyer and Joshi, 1999; Joshi et al., 2003). This means that the hierarchical structure internal to each elementary tree need not be reproduced in the semantic representation. This leads to the so-called *flat* semantic representation, that is, the semantic expression associated with the sentence is essentially a conjunction of semantic expressions associated with each elementary tree.[5] Of course, relevant machinery has to be provided for scope information (for details, see Kallmeyer and Joshi, 1999). The semantics need not be compositional at the level of the elementary trees. It is, however, compositional at the level of the derivation tree, i,e, at the level at which the elementary trees are assembled. This aspect of the architecture is also helpful in dealing with some of the noncompositional aspects, as in the case of rigid and flexible idioms (see Abeillé, 2002, chap. 1; Stone and Doran, 1999).

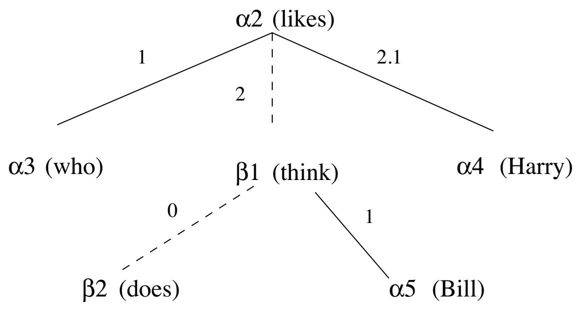


FIGURE 1.11 LTAG derivation tree

1.4 Supertagging

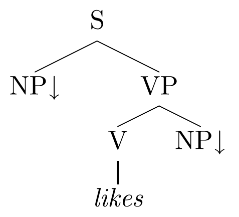
The elementary trees associated with a lexical item can be treated as if they are more informative parts-of-speech (super POS (parts-of-speech) or supertags) in contrast to the standard POS such as V (verb), N (noun), and so on. Now, it is well known that local statistical techniques can lead to remarkably successful disambiguation of standard POS. Can we apply these techniques for disambiguating supertags, which are very rich descriptions of the lexical items? If we can, then, indeed, this will lead to *almost* parsing. This approach is called supertagging (Joshi and Srinivas, 1994b; Srinivas and Joshi, 1998).

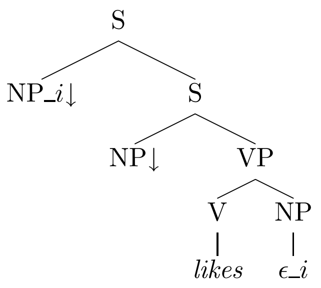
In figure 1.12, two elementary trees associated with the lexical item *likes* are shown. These are the same trees we have seen before. However, now we are going to regard these trees as super part-of-speech (supertags) associated with *likes*. Given a corpus parsed by LTAG grammar we can compute the statistics of supertags, statistics such as unigram, bigram, and trigram frequencies. Interestingly, these statistics combine not only lexical statistics but the statistics of constructions (as represented by the elementary trees) in which the items appear, thus combining lexical statistics with the statistics of the linguistic environments in which the lexical items appear.

Thus, for example, consider the string

The purchase price includes two ancillary companies

as shown in figure 1.13. The supertags associated with each word appear on top of that word. Some words have only only one supertag associated with them and others have more than one. In the current system there are about 15 to 20 supertags per word on the average, so there is a very high level of local ambiguity. In figure 1.14, the same supertags are shown for each word; however, for each word one supertag has been identified (in a box). This is the *correct* supertag for this word in the sense that this is the supertag associated with this word in the correct parse of this sentence. Suppose we are able to find the correct supertag for each word in this sentence by





transitive object extraction

Some other trees for *likes*: subject extraction, topicalization, subject relative, object relative, passive, and so on.

FIGURE 1.12 Two supertags for *likes*

applying local statistical disambiguation techniques; then for all practical purposes we will have parsed the sentence. It is not a complete parse because we have not put the supertags together; hence we call it an *almost* parse.

A supertagging experiment was carried out using trigrams of supertags and techniques similar to the standard POS disambiguation techniques (Joshi and Srinivas, 1994a). The corpus used was the Wall Street Journal Corpus (WSJ). With a training corpus of 1 million words and a test corpus of 47,000 words, the baseline performance was 75% (that is, 75% of the words received the correct supertag). The baseline corresponds to the case when the supertag chosen for a word is just the most frequent supertag for this word. We know from the performance of disambiguators for the standard POS that the baseline performance is 90% or better. The lower baseline performance for supertagging is due to the fact that the local ambiguity is very high (about 15 to 20 on the average) in contrast to the local ambiguity of standard POS, which is about 1.5 for English. The performance of the trigram supertagger, on the other hand, is 92%. The improvement from 75% to 92% is indeed very remarkable. This means that 92% of the words received the correct supertag. More recent experiments based on other machine learning techniques have pushed the performance to about 93% (Chen and Vijay-Shanker, 2000; Shen and Joshi, 2003).[6]

Of course, more can be said about this supertagging approach. There are techniques to improve the performance and to make the output look more like a complete parse. We will not discuss these aspects; rather, we will talk about the abstract nature of supertagging and its relevance to the use of the CLSG approach. In supertagging we are working with complex (richer) descriptions of primitives (lexical items in our case). The descriptions of primitives (lexical items in our case) are complex because we try to associate with each primitive all

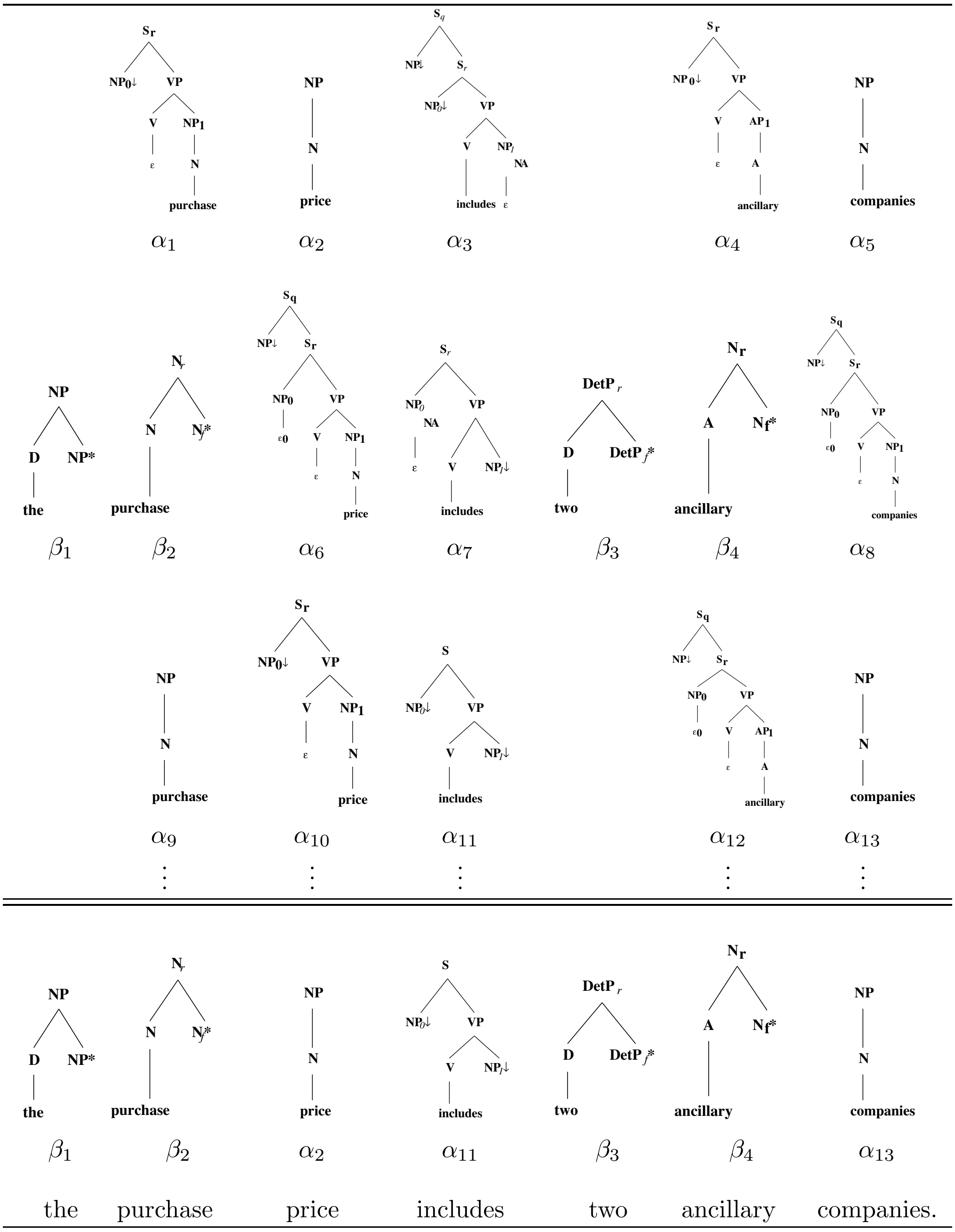
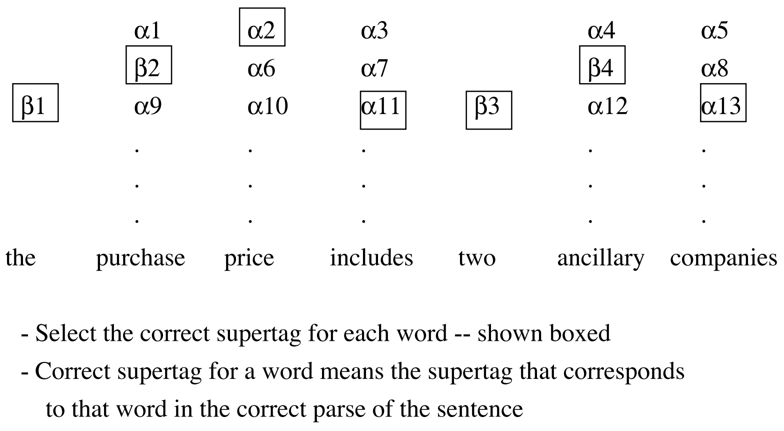


FIGURE 1.13 A selection of the supertags associated with each word of the sentence: *the purchase price includes two ancillary companies*



α1
α2
α3
α4
α5
β2
α6
α7
β4
α8
β1
α9
α10
α11
β3
α12
α13
the
purchase
price
includes
two
ancillary
companies
- Select the correct supertag for each word -- shown boxed
- Correct supertag for a word means the supertag that corresponds to that word in the correct parse of the sentence

FIGURE 1.14 A sentence with the correct supertag for each word

information relevant to that primitive. Making descriptions more complex has two consequences: (1) local ambiguity is increased, that is, there are many more descriptions for each primitive, however, (2) these richer descriptions of primitives *locally* constrain each other. There is an analogy here to a jigsaw puzzle—the richer the description of each piece the better, in the sense that there are stronger constraints on what other pieces can go together with a given piece. Making the descriptions of primitives more complex allows us to compute statistics over these complex descriptions but, more importantly, these statistics are more meaningful because they capture the relevant dependencies directly (for example, word-to-word dependencies where each word is the lexical anchor of some supertag, and word-to-construction dependencies). Local statistical computations over these complex descriptions lead to robust and efficient processing. Supertagging by itself is not full parsing. However, parsing a sentence already supertagged is far more efficient (faster), on the average, as compared to parsing without supertagging. Supertagging is thus an example of a local computation on complex descriptions. Psycholinguistic relevance of supertagging is described in section 1.5.

These considerations are directly relevant to other domains, such as AI. We can illustrate this by pointing out interesting relationships to the well-known algorithm in Waltz (1975) for interpreting line drawings. What Waltz did was to make the descriptions of vertices more complex by adding information about the number and types of edges incident on a vertex. Again, there is an analogy here to a jigsaw puzzle: the richer the description of a piece the better. By making the descriptions of vertices more complex, the local ambiguity was increased, for example, an L junction (a particular kind of junction in the taxonomy of junctions of edges) has about 92 physically possible labelings. However, local computations on these complex descriptions are adequate to rapidly disambiguate these descriptions, leading to efficient computation.

Data Oriented Parsing (DOP) (Bod et al., 2003) is another framework that uses richer descriptions that extend the domain of locality of constraints. In this approach, *all* possible cuts of a parse tree are maintained and analysis of a sentence proceeds by pasting these structures together and computing the most probable derivation. The structures, however, need not be lexicalized, and the tree cuts would typically need not result in a linguistically meaningful structure. Along the lines of extending the domain of locality of CFG rules, research has been carried out on using richer (tree-local) features in machine learning techniques applied to statistical natural language parsing (Magerman and Marcus, 1991; Black et al., 1993; Magerman, 1995; Collins, 1996; Ratnaparkhi, 1997; Charniak, 2000). These features might be regarded as a bundle of constraints anchored on a lexical item just as supertags. Arbitrary feature co-constraints can be computed using tree-kernels as shown in Collins and Duffy (2002). However, in these approaches *recursion* is not factored out from the domain over which the constraints operate. As a result, feature contexts that only differ due to recursive elements are not identified leading to a large number of contexts and consequently to sparseness of data and lack of model portability issues.

1.5 Supertags in Psycholinguistic Models

In this section, we will discuss the implications of the TAG architecture for certain processing issues. These pertain to using supertags to make fine-grained distinctions between lexical and structural ambiguities and their relevance to processing.

Recently there has been increasing convergence of perspectives in the fields of linguistics, computational linguistics, and psycholinguistics, especially with respect to the representation and processing of lexical and grammatical information. More specifically, this convergence is due to a shift to lexical and statistical approaches to sentence parsing. The particular integration of lexical and statistical information proposed in Kim et al. (2002) is highly relevant from the perspective of the LTAG architecture. As we saw in section 1.3, LTAG associates with each lexical item one or more elementary structures, which encapsulate the syntactic and associated semantic dependencies. The computational results in supertagging as described earlier in section 1.4 show that much of the computational work of linguistic analysis, which is traditionally viewed as the result of structure building operations can be viewed as lexical disambiguation in the sense of supertag disambiguation. If the supertagging model is integrated in a psycholinguistics framework, then one would predict that many of the initial processing commitments of syntactic analysis are made at the lexical level in the sense of supertagging. The model proposed in Kim et al. (2002) is an integration of the Constraint-Based Lexicalist Theory (CBL) (MacDonald et al., 1994), where the lexicon is represented as supertags with their distributions estimated from corpora as in the supertagging experiments described earlier (section 1.4).

For example, in this model, there is a distinction between the prepositional phrase attachment ambiguity (PP ambiguity) as in (1) below

(1) *I saw the man with a telescope*

and the PP attachment ambiguity as in (2) below

(2) *The secretary of the general with red hair*

In the first case, the PP either modifies a noun phrase, *the man*, or a verb phrase VP, headed by *saw*. There are two supertags associated with the preposition *with*, as in figure 1.15, one with the foot and root nodes being NP (supertag $\beta 1$) or both being VP (supertag $\beta 2$). That is, the ambiguity is resolved if we pick the correct supertag for *with* anchored on the preposition *with*. Thus, this PP attachment ambiguity will be resolved at the lexical level. However, in the second case in both readings of (2) the supertag associated with *with* is the one whose root and foot nodes are both NP. Thus, in this case, the ambiguity will not be resolved at the lexical level. It can only be resolved at the level when the attachment is computed.

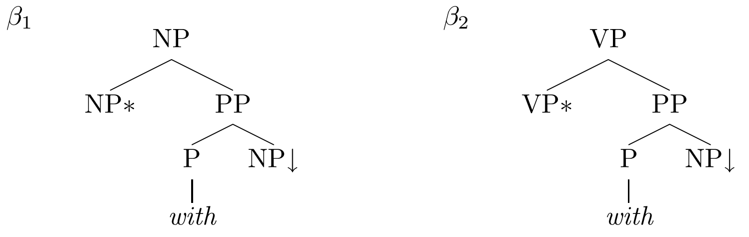


FIGURE 1.15 Two supertags for *with*

The first PP attachment ambiguity is not really an attachment ambiguity. It should be resolved at an earlier stage of processing. In the second case it will be resolved at a later stage. Similarly, the ambiguity associated with a verb such as *forgot*, because it can take either an NP complement as in (3) below

(3) *The student forgot her name*

or a VP complement as in (4) below

(4) *The student forgot that the homework was due today*

is a lexical (supertag) ambiguity and need not be viewed as a structural ambiguity. Kim et al. (2002) present a neural net based architecture using supertags and confirm these and other related results.

1.6 Outline of the Book

As discussed in the preceeding sections, the perspective of viewing the elementary trees of LTAG as supertags has provided novel and interesting insights into computational and psycholinguistic approaches to language processing. During this past decade, this theme has been explored in different grammar formalisms, and the consequences of such a localization for grammar development as well as natural language applications have been extensively studied. This book is a collection of some of the research that investigates this theme. We expect that this book will be of special interest to computational linguists and researchers in speech and language processing for its perspective on the representation and its consequences on computation of linguistic structure. For the machine learning community interested in language applications, we expect this book to provide an opportunity for exploring novel machine learning techniques that can exploit the richer feature space provided by supertag representations. The close coupling of lexical and syntactic information afforded by the supertag representation and its impact on language processing would be of interest to researchers of psycholinguistics interested in human sentence processing.

The book is broadly organized into five parts. The first part highlights issues related to supertags in LTAGs such as the creation and organization of supertags. Research concerning the models for supertag disambiguation and their relation to parsing in the context of LTAG supertags constitute the second part of the book. The third part presents different instantiations of the notion of supertags in a range of grammar formalisms. Research work on some of the linguistic and psycholinguistic issues related to supertags constitute the fourth part. And finally, some of the speech and language applications that exploit supertags are highlighted in the last part of the book.

1.6.1 Developing and organizing supertags

The construction of wide-coverage grammars had been a dominant activity in the natural language processing community during the early nineties. Broad coverage grammars and parsers were constructed in HPSG (Flickinger et al., 2000), XTAG (XTAG-Group, 2002), LFG (Butt et al., 2002). These grammar development projects involved construction of detailed analysis of linguistic phenomena and encoding them in the concerned formalisms. At the same time, parse-annotated corpora such as the Penn Treebank (Marcus et al., 1993) and NEGRA (Skut et al., 1997) were being created. Although the representations used in these treebanks were relatively shallow compared to the hand-built grammars, they provided a distributional characterization of the linguistic phenomena as evidenced in the domain from which the corpus was drawn. The two chapters in this part of the book discuss different methods for creating and organizing an inventory of supertags that rely on parsed corpora as well as hand-crafted grammars.

In chapter 2, Xia and Palmer describe methods used to transform the parse-annotated Penn Treebank corpus into a corpus of LTAG derivations and the associated set of supertags. Such an LTAG corpus has been used for training supertagging models as well as statistical LTAG parsers. This chapter also discusses the issues related to the extraction of an LTAG grammar in contrast to extracting a CFG; a key difference is the need to distinguish arguments from adjuncts in an LTAG grammar. The chapter provides coverage statistics of the extracted grammar, results of supertagging experiments, using the extracted supertag set as well as the relation of this work to similar grammar extraction methods.

An alternate method to constructing the inventory of supertags is presented in the *LexOrg* system discussed in chapter 3. The motivation for this work is to factor out the structural redundancy present in the set of supertags and instead to represent supertags as structures created by composition of tree descriptions each of which are associated with some linguistic aspect such as subcategorization, head-modifiers, syntactic variations (e.g., passivization, relativization). The tree descriptions are represented as formulae in a simplified first-order language and the model that satisfies these formulae results in the desired set of supertags. This approach not only compacts the grammar but also provides an abstract specification for the supertags based on linguistic principles. This abstract specification of supertags directly allows for rapid creation of supertag sets for new languages. This chapter discusses the creation of supertag sets for Korean and Chinese languages and compares them to the English supertag set.

1.6.2 Supertagging and parsing

The chapters in this part of the book explore different models for supertagging and explicate the relationship between supertagging and parsing.

In chapter 4, Satta reviews lexicalized formalisms such as lexicalized CFG and TAG. The focus of the chapter is a formal presentation of parsing algorithms for such lexicalizd formalisms. The crucial observation is that using algorithms designed for parsing unlexicalized grammars to parse such lexicalized formalisms leads to inefficient parsing algorithms. As a result of increase in the number of nonterminals due to lexicalization, the complexity of parsing increases by a factor of n^2 resulting in $O(n^5)$ parsing complexity for binary branching lexicalized context-free grammars(LCFG). The chapter reviews an algorithm that reduces the complexity of LCFG parsing to $O(n^4)$ and extends this same algorithm to lexicalized TAGs resulting in a reduction of complexity from $O(n^8)$ to $O(n^7)$. In all these complexity results there is an additional constant factor that is cubic in the number of grammar primitives. This constant factor typically dominates parsing performance in broad-coverage natural language grammars. The supertagging approach helps in reducing this constant factor significantly and improves the speed of the parser dramatically.

Sarkar, in chapter 5, presents two interesting research directions concerning the use of supertagging. First, the chapter provides quantitative results on how a lexicalized TAG parser's efficiency is dependent on syntactic lexical ambiguity and sentence complexity (defined in terms of the number of clauses in a sentence). The use of supertagging before parsing is shown to dramatically reduce the syntactic lexical ambiguity and radically improve parsing efficiency. A second line of research presented in the chapter concerns with using the co-training paradigm to bootstrap parse-annotated corpora. In the co-training paradigm for parsing, two statistical parsers assign probability scores to their input based on (ideally) conditionally independent features and produce the same parse output. Given unannotated sentences, the high scoring parses from one parser are used as training data for the other parser, thus the two parsers mutually benefit from a growing size of parse annotated corpus. The chapter presents improvement results in parsing accuracy using a small amount of annotated corpus in conjunction with two parsers – a supertagger-based parser and a statistical LTAG parser.

In chapter 6, Shen explains the use of discriminatively trained classification techniques for supertag disambiguation. The approach described in this chapter relies on building one classifier for each part-of-speech that predicts the supertag label given local contextual information such as lexical context, the preceeding supertag predictions, and the part-of-speech tags. This rich conditioning information is used to build a log-linear classification model. He further combines the supertags from a left-to-right disambiguation model with a right-to-left disambiguation to improve the supertagging accuracy results. In the second part of the chapter, he contrasts supertags to part-of-speech tags and shows that the richer and detailed information encoded in the supertags is indeed beneficial in improving the accuracy on the task of chunking nonrecursive noun phrases.

Models for supertagging have typically relied on estimating statistical models from annotated corpora. In chapter 7, Boullier presents an alternate approach to supertagging that exploits the structural constraints of the underlying LTAG grammar and does not rely on a statistical model for disambiguation. A drawback in statistical models of supertagging, both one-best and n-best variants, is that they might eliminate supertags that are necessary for parsing a sentence. The approaches presented in this chapter guarantee that the correct supertag will not be eliminated from the set of supertags assigned to a word. Boullier presents different supertaggers that are modeled using context-free and regular approximations of the LTAG grammar. The chapter details the construction of the supertaggers and evaluates the different approximations in terms of their precision of supertag assignment on the Wall Street Journal corpus. The speed versus precision of the supertaggers are also explored. An interesting approach of combining the nonstatistical supertagger, which has 100% recall with a statistical supertagger to improve precision is also suggested.

Nasr and Rambow in chapter 8 present a nonlexicalized chart parser (GDG) that uses the output of a supertagger and produces a dependency parse output for a sentence. In contrast to the Lightweight Dependency Analyser(LDA) (Bangalore, 2000) that uses heuristics to compute links between supertags, the GDG parser uses probabilities estimated from a parsed corpus to compute the dependency linkage structure. Also, being a full parser, unlike LDA, a globally consistent linkage structure is produced. The grammar is encoded as an Recursive Transition Network (RTN), where each finite-state automaton represents a supertag and a transition is labeled by all supertags that can substitute into that supertag at a given node. Given the supertag sequence or the n-best output from the supertagger, the chart parser uses the RTN to produce a dependency parse forest from which the first-best dependency parse is retrieved. The parser is evaluated on the Penn Treebank and different trade-offs in terms of the supertag ambiguity versus the parser efficiency and accuracies are explored. There is also a detailed discussion in the chapter on how this research is different from other recent work in dependency parsing.

1.6.3 Supertags in other related formalisms

The adoption of the notion of supertags by researchers working in different grammar formalisms is the topic of the third part of the book. Although, as one would expect, the supertags in these different formalisms result in different representations, the notion of localized structures provides a unifying theme across the different formalisms. The differences in encodings of linguistic phenomena in these formalisms could be an object of future study in its own right.

Combinatory Categorial Grammars (CCG) and Lexicalized Tree-Adjoining Grammars are perhaps the two most closely related lexicalized grammar formalisms that have been extensively compared from formal, computational and linguistic perspectives. In chapter 9, Clark and Curran extend this tradition by presenting an approach to supertagging using CCG categories and tightly integrating the supertagger with a CCG parser. Clark and Curran present a maximum entropy model for CCG supertag disambiguation which is extended to produce multiple CCG supertags per word. The tight coupling with the parser is novel in the sense that the parser requests the supertagger more supertags if the parser fails to produce a spanning analysis. Clark and Curran use a CCG grammar extracted from the Penn Treebank and train and evaluate the supertagger and parser on standard partitions of the treebank. The performance of the CCG supertagger is better than that reported for LTAG supertagger partly due to the size of the supertag sets in these two grammars. CCG supertag sets are usually smaller than the LTAG supertag sets due to the difference in granularity of localization in the two grammars. They also crucially rely on supertagging to reduce the search space during the discriminative training phase of the CCG parser.

Constraint Dependency Grammars introduced by Maruyama (1990) associates lexical items with *governor* and *need* roles that are filled in by role values that indicate the dependency relations among the words of a sentence. A parse structure in these grammars is a consistent assignment of role values to the need roles. A set of constraints operating on the lexial level (fullfillment of need roles) and at the sentence level (e.g., each word must have a single governor) are used to specify the consistency of a parse. The search for a parse proceeds as a consistraint satisfaction program. In chapter 10, Harper and Wang extend this grammar framework to accomodate for ambiguity in lexical categories (SuperARVs) and incorporate probabilistic constraints in order to model parse preferences. The SuperARVs (super abstract role values) are lexicalized representations and encode rich morpho-syntactic information. However, unlike supertags, SuperARVs do not contain constituency information. Broad-coverage statistical parsers using SuperARVs extracted from the Penn Treebank are presented in this chapter. Parsing using the two methods of (a) SuperARV disambiguation followed by dependency linking and (b) tight integration of SuperARV disambiguation as part of dependency linking are presented and compared using the Wall Street Journal parsing task. They also present comprehensive results on large vocabulary speech recognition and demonstrate the use of structural information as encoded in the SuperARV grammars improve speech recognition accuracy as compared against n-gram based language models.

The trade-off between the amount of information that is packed into supertags (granularity of supertags), the accuracy of supertagging and its relation to parsing accuracy is investigated in chapter 11. Foth and colleagues investigate the effect of incorporating ever richer information into supertags. These supertags are constructed from dependency trees for German that were extracted from the NEGRA and TIGER corpora. The obvious effect of incorporating richer information into supertags is an increase in the supertag vocabulary. However, interestingly, this increase in supertag vocabulary does not directly correlate with the supertagging error. They show that supertagging using a supertag set that includes direction of attachment information in a supertag is harder to dismabiguate than a supertag set which includes complement information in a supertag. Further, the disambiguated supertag sequence is encoded as weighted constraints and incorporated into a rule-based weighted constrained dependency grammar parser. The evaluation of parser output presented in this chapter illustrates that increasing the complexity of supertags and using the disambiguated supertags to guide the parser continues to improve the accuracy of the parser. The information encoded in the supertags counter balances the drop in accuracy of supertag disambiguation with more complex supertags.

Moot, in chapter 12, describes type-logical grammars, which trace their lineage to the Adjukiewicz-Bar-Hillel grammars. The analysis/generation of a sentence in type-logical grammars is viewed as a proof using rules of natural deduction. In order to account for linguistic phenomena that require more than context-free generative capacity, modal operators that allow for structural changes of the proof tree are introduced in this chapter. Following an introduction to type-logical grammars, Moot introduces the idea of extracting a type-logical grammar from a dependency parse treebank. He extracts a grammar from a parsed corpus of *spoken* Dutch sentences. The grammar extraction procedure is general enough that it could be directly applied to other dependency annotated treebanks. The grammar, as in LTAG, is lexicalized and by varying the information present in the type-logical supertags, Moot shows that the supertag disambiguation can be improved. Given the complexities of the spoken corpus – disfluencies and ellipses, the supertag set is quite large compared to other supertag sets and the supertagging accuracy is correspondingly lower as well.

In chapter 13, Neumann and Crysmann discuss results on statistical parsing using a richly annotated German treebank from the Verbmobil domain. The treebank is annotated using a Head-driven Phrase Structure Grammar representation. They discuss the extraction procedure that exploits the head/argument and argument/adjunct distinctions as defined by the HPSG grammar. The result of the grammar extraction is a lexicalized tree-insertion grammar (LTIG) and the lexicalized trees are used as HPSG-supertags in a probabilistic LTIG parser. They report results on parsing German sentences from the Verbmobil domain and contrast them to previous results obtained on the NEGRA corpus.

Supertags from the LTAG, CCG, and other formalisms can be viewed as localizing different kinds of linguistic information into a single elementary object. The nature and amount of information that is localized vary depending on the formalism and the linguistic analysis that is adopted in that formalism. It can be regarded as extending the domain of locality of CFG rules by operating on objects richer than single level trees. Matsuzaki, Miyao, and Tsujii present an interesting variant to the question of how to extend the localization of CFG in chapter 14. In their approach, a CFG nonterminal is enriched with latent annotations that take on specified range of values. These latent annotation variables and the values do not bear any linguistic significance, but are used to losen the independence assumptions in a context-free grammar (a motivation shared by LTAGs). A special case of this approach is the use of head-word annotation on CFG nonterminals that has been successfully exploited in statistical parsing research. The chapter discusses in detail the methods for estimating the parameters of PCFGs with latent annotations and presents parsing evaluation results as the number latent variables and the value ranges are varied.

In chapter 15, Bharathi and Sangal discuss a computational framework for Indian language processing based on Paninian Grammar. Panini had formulated a theory for Sanskrit language analysis some two thousand five hundred years ago which has been since extended and adapted for analysis of other Indian languages. This chapter discusses the adoption and suitability of Paninian theory for computational analysis of Indian languages particularly due to their relatively free word order nature. The use of *karaka* roles helps in interfacing the syntax and semantic level of representations. Post-position markers help identify word groups that satisfy the karaka roles and are central to the grammaticality of a sentence. Each verb is associated with karaka frames that specify mandatory and optional karakas. The parsing problem is framed as an integer programming problem and an efficient solution is computed using the bipartite graph-matching algorithm. The resulting parse is a dependency tree similar to a derivation tree in LTAG, but with words as nodes and karaka roles as the labels for the edges of the dependency tree. The similarities between LTAG and Computational Paninian Grammar is explicated and the use of supertagging as a means of selecting the appropriate karaka frame is suggested.

1.6.4 Linguistic and psycholinguistic issues

As discussed in the introductory sections of this chapter, the effect of localization of dependencies to be specified in a supertag has interesting linguistic and psycholinguistic implications. In the fourth part of the book, we include work from two authors who have novel proposals for syntactic analysis that are motivated by representational (syntax-prosody interface) and human sentence processing issues.

Frank (1992) first enunciated the principles that have been used as a guide in the construction of elementary trees for a linguistically motivated grammars in LTAG. The lexical head of an elementary tree is to assign a role to all the frontier nonterminal nodes in the tree. This limits the size of the elementary tree and consequently all the modifiers are factored out from this domain of role assignment. In chapter 16, Frank restates the principle to assert that *all* the syntactic relations of a lexical head are expressed in an elementary tree and questions the need for construction of a global syntactic structure. In this view, the assignment of the correct elementary structure to words would be sufficient to infer syntactic information of each word – much like the motivation underlying supertagging. In this chapter, Frank explores the consequences of not computing a global syntactic structure for deriving phonological and semantic representations. He proposes that the phonological representations are computed locally for each elementary tree and are combined using a merge operation to form a phonological representation for a sentence.

The relevance of lexicalized grammars for computational analysis of language has been explored extensively. Lexicalized grammars have also been studied as appealing representations in the human sentence processing literature (Trueswell and Tanenhaus, 1994). The domain of locality provided by supertags combined with their distributional information derived from corpora has been shown to model the relevant human sentence processing results related to processing preferences and difficulties (Kim et al., 2002). In chapter 17, Mazzei, Lombardo and Strut explore the issue of incremental sentence processing using supertag-based elementary structures. In order to address the apparent lack of delays in structure computation as evidenced from experimental literature, they propose a *strong connectivity hypothesis*. According to this hypothesis, during left to right sentence processing, each word is incorporated into the evolving syntactic structure immediately. Such a proposal requires extending the domain of locality of supertags further in order to allow for the incorporation of a word into the structure computed for the fragment of a sentence thus far. Also, they develop a dynamic version of TAG called DVTAG and introduce direction (left, right) sensitive substitution and adjunction opertation in order to combine the supertag structures. The chapter also describes the extraction of these larger supertags from annotated corpora and illustrates the size of such extracted supertag sets. The use of such supertag sets in a parser is left for further study.

1.6.5 Speech and language applications

The level of syntactic representation created by the supertagger has been exploited in a variety of speech and language applications. The enriched tagset is used for information filtering (Chandrasekhar and Bangalore, 2002), for language modeling (Srinivas, 1996), and for coreference resolution (Srinivas and Baldwin, 1996). Other interesting speech and language applications that exploit the supertag representation for semantic role labeling, spoken language understanding in dialog and mechanisms for input in devices without keyboard are presented in this final section of the book.

In chapter 18, Chen presents different methods for semantic role labeling using deep linguistic features and compares their performances against methods that use only surface linguistic features. Semantic role labeling is the task of identifying predicate-argument relations and labeling the arguments of the predicates. These labels are as defined in the Propbank (Penn Treebank with semantic role labels), annotation guide and represent deep grammatical roles, in contrast to the surface grammatical roles of subject, direct and indirect object. The chapter builds on the previous work of extracting LTAG grammars from the Penn Treebank. With the availability of Propbank, the extracted grammars would contain the role label as part of the syntactic constituent label. The extracted LTAG grammars are used to build supertaggers as well as lightweight dependency analyzers for semantic parsing. This approach is contrasted against a full statistical semantic parsing approach. The conclusions of these experiments indicate that supertags capture most of the deep linguistic information that improve the semantic role labeling task performance. Also, the use of semantic supertags (semantic roles on argument nodes) improves the performance over an approach of using a syntactic parser and then recovering the semantic role labels as a second step.

In chapter 19, Harbusch and colleagues discuss two other applications for supertagging. In the first application, the supertagger is used in the understanding component of a user-initiated dialog system in a call center. They show that the use of the supertagger enhances the robustness of the spoken language understanding component in terms of classifying the user's request as well as extracting the task parameters more accurately. The second application presented in this chapter concerns ambiguous keyboards and multitapping on small devices without a conventional keyboards. These keyboards cluster characters onto single keys and the user needs to tap multiple times to select the appropriate character. The method proposed in this chapter involves allowing the user to tap only once on the multitap keyboard and to resolve the character ambiguity generated by the single tap using sentence-based language models. The supertagger is used to provide syntactic constraints for the reordering of the sentence hypotheses and allows the user to select the appropriate sentence from the ranked suggestion list. The evaluation results support the use of supertagger in improving the interaction time compared to the word-based disambiguation methods.

1.6.6 Ongoing research related to supertags

There are several research directions related to supertags that are not covered by the topics of the chapters included in this book. We summarize these briefly in this section.

The research work on discriminative models for supertagging in conjunction with efficient statistical tree-insertion grammar parsing is available in MICA, a broad-coverage, fast English parser. The parser can be downloaded freely (Bangalore et al., 2009).

While most of the supertag disambiguation discussed in this book are for sentence analysis, there has been work on using them for *sentence generation*. Sentence generation is typically viewed as consisting of content planning, sentence planning and surface realization steps (Reiter and Dale, 2000) which are usually hand-crafted for a given application. The content planning step decides on what is to be conveyed and structures the content for the sentence planner. The sentence planner uses lexical and syntactic resources in order to convey the content. The result of sentence planning is then used by the surface realizer to create a well-formed natural language sentence. In recent work (Langkilde and Knight, 2000; Corston-Oliver et al., 2002; Bangalore and Rambow, 2000) there has been interest in creating a broad-coverage, data-driven surface realizer that could be used across different applications. The approach followed to that end is to regard the surface realizer as transforming an abstract representation (e.g., logical form or underspecified dependency trees) into a natural language sentence and to use statistical models to achieve this transformation. The supertag-based approach to a surface realizer followed in FERGUS (Bangalore and Rambow, 2000) uses underspecified (unlabeled) dependency trees as input. The nodes of the dependency tree are then annotated with supertag labels which provide the lexical ordering information. Assigning supertags to nodes of the dependency tree requires supertagging, just as in the case of sentence analysis. Generative and discriminative models for supertagging have been explored for this task (Bangalore and Rambow, 2000, 2005).

Supertags have been used in a variety of different speech and language tasks as a means of incorporating syntactic information in a inexpensive manner without incurring the cost of a full parse. Supertags have recently been used to improve the quality of phrase tables in phrase-based statistical machine translation research (Hassan et al., 2007). The authors show that assigning supertags to words of a phrase and exploiting the constraints encoded in supertags helps in improving the translation quality over a purely lexical phrase-based approach.

For speech prosody prediction, supertags have been incorporated along with acoustic information in a maximum-entropy framework (Rangarajan et al., 2007). The supertag labels are shown to provide discriminative information that improves the prosody label prediction beyond the lexical information of a sentence. Furthermore, supertags complement the acoustic information of the speech signal as well and a combination of lexical, syntactic and acoustic information is shown to perform the best in this task. A similar result is shown for dialog act labeling, where each utterance of a dialog turn is assigned a communicative intent label. Here too, exploiting the supertag label in addition to the lexical and acoustic information improves labeling accuracy significantly (Bangalore et al., 2006; Rangarajan et al., 2007).

Finally, LTAG and CCG formalisms continue to be compared on formal, linguistic, and computational grounds. While the work of viewing CCG categories as richer lexical description has been explored in chapter 9, there is also work on extending the notion of CCG categories to be closer to the notion of supertags. Supertags associated with a predicate encode not only the arguments of a predicate but also all the different structural positions the arguments may occupy in various syntactic constructions in which the predicate participates. Categorial grammars also encode argument positions, but they do not encode (in the representation of a category label associated with a predicate) the different structural positions that the arguments can occupy. These emerge during the various types of compositions and type-raisings in categorial grammars. In this sense, LTAGs are strongly lexicalized. Hence although the so-called Combinatory Categorial Grammars (CCG) (Steedman, 1996) are weakly equivalent to LTAGs (both belonging to the class of the so-called *mildly context-sensitive languages* (Joshi, 1985), they are not strongly equivalent. It is possible to construct a version of CCG that is more like LTAG. This is achieved by starting with the syntactic type assigned to a predicate by a CCG and then *unfolding* it and creating partial proof trees, analogous to the supertags in LTAG, which are then composed by appropriate rules of composition (inference), for further details see Joshi and Kulick (1997).

Notes

1. The Greibach form of the rule is related to the categories in a categorial grammar.

2. In the actual LTAG grammar, each node in an elementary tree is decorated with attribute value structures (feature structures) that encode various linguistic constraints specified over the domain of an elementary tree. There is no recursion in these feature structures. We omit these details here in as much as they are not essential for our immediate purpose.

3. This distinction between the two types of elementary trees is characterized in LTAG in terms of *initial trees* (the α trees) and the *auxiliary trees* (the β trees).

4. The derivation trees of LTAG have a close relationship to dependency trees, although there are some crucial differences. The semantic dependencies are the same, however.

5. The general notion of flat semantics is related to the notion of minimal recursion semantics (MRS) (Copestake et al., 1999). MRS has been used in the semantic computation in the HPSG framwork. In the LTAG framework, the notion of elementary trees and the derivation tree, which specifies the composition in terms of the elementary trees, directly provides a representation for computing a *flat* semantics.

6. There have been several recent experiments using different methods for extracting supertags from treebanks. Each of these extraction methods results in different sizes of supertag tagsets and hence the accuracy of the supertag disambiguation varies depending on the supertag tageset.

References

Abeillé, A. (2002). Une grammaire électronique du français. *CNRS Éditions.*

Bangalore, S. (2000). A lightweight dependency analyzer for partial parsing. *JNLE*, 6(2):113–138.

Bangalore, S., Boullier, P., Nasr, A., Rambow, O., and Sagot, B. (2009). MICA: A probabilistic dependency parser based on tree insertion grammars (application note). In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 185–188.

Bangalore, S., Fabbrizio, G. D., and Stent, A. (2006). Learning the structure of task-driven human-human dialogs. In *Proceedings of COLING/ACL.*

Bangalore, S. and Rambow, O. (2000). Exploiting a probabilistic hierarchical model for Generation. In *COLING*, Saarbucken, Germany.

Bangalore, S. and Rambow, O. (2005). Classification of structured descriptions. In *Proceedings of ICASSP*, Philadelphia.

Black, E., Jelinek, F., Lafferty, J., Magerman, D. M., Mercer, R., and Roukos, S. (1993). Towards History-based Grammars: Using Richer Models for Probabilistic Parsing. In *Proceedings of the* 31^{st} *Conference of Association of Computational Linguistics.*

Bod, R., Scha, R., and Sima'an, K. (2003). *Data-Oriented Parsing.* CSLI.

Butt, M., Dyvik, H., Holloway-King, T., Masuichi, H., and Rohrer, C. (2002). The parallel grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation.*

Chandrasekhar, R. and Bangalore, S. (2002). Glean: Using syntactic information in document filtering. In *Encyclopedia of Microcomputers*, pages 111–131. Marcel Dekker.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of NAACL.*

Chen, J. and Vijay-Shanker, K. (2000). Automated extraction of the tags from the penn treebank. In *Proceedings of the* $6^t h$ *International Workshop on Parsing Technology (IWPT)*, pages 65–76.

Chiang, D. (2000). Statistical parsing with an automatically extracted tree adjoining grammar. In *Proceedings of the Association for Computational Linguistics (ACL) 2000 Meeting.*

Collins, M. (1996). A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the* 34^{th} *Annual Meeting of the Association for Computational Linguistics*, Santa Cruz.

Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures and the voted perceptron. In *ACL*.

Copestake, A., Flickinger, D., Sag, I. A., and Pollard, C. (1999). Minimal recursion semantics: An introduction. Technical report, Stanford University.

Corston-Oliver, S., Gamon, M., Ringger, E., and Moore, R. (2002). An overview of amalgam: A machine-learned generation module. In *Proceedings of INLG-02*.

Flickinger, D., Copestake, A., and Sag, I. A. (2000). HPSG analysis of English. In Wahlster, W., ed., *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 254–263. Springer Verlag.

Frank, R. (1992). *Syntactic locality and Tree Adjoining Grammar: grammatical, acquisition and processing perspectives*. PhD thesis, University of Pennsylvania, IRCS-92-47.

Hassan, H., Sima'an, K., and Way, A. (2007). Integrating supertagging into phrase-based statistical machine translation. In *Proceedings of ACL 2007*, Prague.

Joshi, A. K. (1985). Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions. In D. Dowty, L. K. and Zwicky, A., eds., *Natural Language Parsing*, pages 206–250. Cambridge University Press.

Joshi, A. K., Kallmeyer, L., and Romero, M. (2003). Flexible composition in LTAG: Quantifier scope and inverse linking. In *Proceedings of the International Workshop on Computational Semantics (IWCS-5)*, Tilburg.

Joshi, A. K. and Kulick, S. (1997). Partial proof trees as building blocks for a categorial grammar. *Linguistics and Philosophy*.

Joshi, A. K., Levy, L. S., and Takahashi, M. (1975). Tree adjunct grammars. *Journal of Computer and System Sciences*, 10:1:136–163.

Joshi, A. K. and Schabes, Y. (1997). Tree-adjoining grammars. In Rosenberg, G. and Salomaa, A., eds., *Handbook of Formal Languages*, pages 69–123. Springer.

Joshi, A. K. and Srinivas, B. (1994a). Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the* 17^{th} *International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan.

Joshi, A. K. and Srinivas, B. (1994b). Disambiguation of super parts of speech (supertags): Almost parsing. In *Proceedings of the 1994 International Conference on Computational Linguistics (COLING)*, Kyoto, Japan.

Joshi, A. K. and Vijay-Shanker, K. (1999). Compositional semantics with lexicalized tree-adjoining grammar (LTAG): How much underspecification is necessary. In Bunt, H. and Thijsse, E., eds., *Proceedings of the Third International Workshop on Computational Semantics (IWCS-3)*, pages 131–145.

Joshi, A. K., Vijay-Shanker, K., and Weir, D. J. (1991). The convergence of mildly context sensitive grammatical formalisms. In Sells, P., Shieber, S., and Wasow, T., eds., *Foundational Issues in Natural Language Processing*. MIT Press.

Kallmeyer, L. and Joshi, A. K. (1999). Factoring predicate argument and scope semantics: Underspecified semantics with LTAG. In *Proceedings of the Twelfth Amsterdam Colloquium, University of Amsterdam*, pages 169–174.

Kaplan, R. and Bresnan, J. (1983). Lexical-functional grammar: A formal system of grammatical representation. In Bresnan, J., ed., *The Mental Representation of Grammatical Relations*. MIT Press.

Kasper, R., Kiefer, B., Netter, K., and Vijay-Shanker, K. (1995). Compilation of HPSG to TAG. In *Proceedings of the Association for Computational Linguistics (ACL)*, MIT Press, pages 92–99.

Kim, A., Srinivas, B., and Trueswell, J. (2002). The convergence of lexicalist perspectives in psycholinguistic and computational linguistics. In Merlo, P. and Stevenson, S., eds., *Sentence Processing and the Lexicon: Formal, Computational and Experimental Perspectives*. John Benjamin Publishing.

Kroch, A. (1989). Asymmetries in long distance extraction in a tree-adjoining grammar. In Baltin, M. and Kroch, A., eds., *Alternative conceptions of phrase structure*. University of Chicago Press.

Kroch, A. and Joshi, A. K. (1985). Linguistic relevance of tree-adjoining grammars. Technical report, Department of Computer and Information Science, University of Pennsylvania.

Kroch, A. and Santorini, B. (1991). The derived constituent structure of the west germanic verb raising constructions. In Freiden, R., ed., *Principles and parameters in comparative grammar*, pages 269–338. MIT Press.

Langkilde, I. and Knight, K. (2000). Forest-based statistical sentence generation. In *Proceedings of First North American ACL*.

Linz, P. (2001). *An Introduction to Formal Languages and Automata*. Jones and Bartlett.

MacDonald, M., Pearlmutter, N., and Seidenberg, M. (1994). Lexical nature of syntactic ambiguity resolution. *Psychological Review*, 101:676–703.

Magerman, D. M. (1995). Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

Magerman, D. M. and Marcus, M. P. (1991). Pearl: A probabilistic chart parser. In *Proceedings of the European Assoc. for Comp. Ling.*, Berlin.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.2:313–330.

Maruyama, H. (1990). Constraint Dependency Grammar and its weak generative capacity. *Computer Software*.

Pollard, C. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University Press of Chicago.

Prolo, C. A. (2003). *LR parsing for tree-adjoining grammars and its applications to corpus based natural language parsing*. PhD thesis, University of Pennsylvania.

Rambow, O. (1994). *Formal and computational aspects of natural language syntax*. PhD thesis, University of Pennsylvania.

Rangarajan, V., Bangalore, S., and Narayanan, S. (2007). Exploiting acoustic and syntactic features for prosody labeling in a maximum entropy framework. In *Proceedings of NAACL 2007*, Rochester.

Ratnaparkhi, A. (1997). A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. In *Proceedings of the Empirical Methods in Natural Language Processing*, New Providence.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.

Resnik, P. (1992). Probabilistic tree-adjoining grammars as a framework for statistical natural language processing. In *Proceedings of COLING '92*, Nantes, pages 418–424.

Sarkar, A. (2002). *Combining labeled and unlabeled data in statistical natural language processing*. PhD thesis, University of Pennsylvania, Philadelphia.

Schabes, Y. (1992). Stochastic lexicalized grammars. In *Proceedings of COLING '92*, University of Chicago Press, pages 426–432.

Shen, L. and Joshi, A. K. (2003). A SNoW based supertagger the applications to NP chunking. In *Proceedings of the Association for Computational Linguistics Meeting (ACL)*, Sapporo, Japan, pages 89–96.

Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1997). An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing ANLP-97*, Washington, DC.

Srinivas, B. (1996). “Almost Parsing” Technique for Language Modeling. In *Proceedings of ICSLP96 Conference*, Philadelphia, USA.

Srinivas, B. and Baldwin, B. (1996). Exploiting supertag representation for fast coreference resolution. In *Proceedings of the International Conference on Natural Language Processing and Industrial Applications (NLP+IA '96)*, Moncton, Canada.

Srinivas, B. and Joshi, A. K. (1998). Supertagging: An approach to almost parsing. *Computational Linguistics*, 22:1–29.

Stabler, E. P. (1997). Derivational minimalism. In Retore, C., ed., *Logical Aspects of Computational Linguistics*, pages 68–95. Springer Verlag.

Steedman, M. J. (1996). *Surface Structure and Interpretation.* MIT Press.

Stone, M. and Doran, C. (1999). Sentence planning as description using tree-adjoining grammar. In *Procceedings of the Association for Computational Linguistics (ACL) Meeting*, Madrid.

Trueswell, J. and Tanenhaus, M. (1994). Toward a lexicalist framework for constraint-based syntactic ambiguity resolution. In Clifton, C., Rayner, K., and Frazier, L., eds., *Perspectives on Sentence Processing.* Lawrence Erlbaum Associates.

Vijay-Shanker, K. (1987). *A Study of Tree-Adjoining Grammars.* PhD thesis, University of Pennsylvania.

Waltz, D. (1975). Understanding line drawings of scenes with shadows. In Winston, P., ed., *The Psychology of Computer Vision.* McGraw Hill.

Weir, D. J. (1988). *Characterizing mildly context-sensitive grammar formalisms.* PhD thesis, University of Pennsylvania.

XTAG-Group (2002). A lexicalized tree-adjoining grammar for English. Technical report, University of Pennsylvania.