Supertagging: An Approach to Almost Parsing

Srinivas Bangalore*
AT&T Labs – Research

Aravind K. Joshi†
University of Pennsylvania

In this paper, we have proposed novel methods for robust parsing that integrate the flexibility of linguistically motivated lexical descriptions with the robustness of statistical techniques. Our thesis is that the computation of linguistic structure can be localized if lexical items are associated with rich descriptions (supertags) that impose complex constraints in a local context. The supertags are designed such that only those elements on which the lexical item imposes constraints appear within a given supertag. Further, each lexical item is associated with as many supertags as the number of different syntactic contexts in which the lexical item can appear. This makes the number of different descriptions for each lexical item much larger than when the descriptions are less complex, thus increasing the local ambiguity for a parser. But this local ambiguity can be resolved by using statistical distributions of supertag co-occurrences collected from a corpus of parses. We have explored these ideas in the context of the Lexicalized Tree-Adjoining Grammar (LTAG) framework. The supertags in LTAG combine both phrase structure information and dependency information in a single representation. Supertag disambiguation results in a representation that is effectively a parse (an almost parse), and the parser need "only" combine the individual supertags. This method of parsing can also be used to parse sentence fragments such as in spoken utterances where the disambiguated supertag sequence may not combine into a single structure.

1. Introduction

In this paper, we present a robust parsing approach called **supertagging** that integrates the flexibility of linguistically motivated lexical descriptions with the robustness of statistical techniques. The idea underlying the approach is that the computation of linguistic structure can be localized if lexical items are associated with rich descriptions (**supertags**) that impose complex constraints in a local context. This makes the number of different descriptions for each lexical item much larger than when the descriptions are less complex, thus increasing the local ambiguity for a parser. However, this local ambiguity can be resolved by using statistical distributions of supertag co-occurrences collected from a corpus of parses. Supertag disambiguation results in a representation that is effectively a parse (**an almost parse**).

In the linguistic context, there can be many ways of increasing the complexity of descriptions of lexical items. The idea is to associate lexical items with descriptions that allow for all and only those elements on which the lexical item imposes constraints to be within the same description. Further, it is necessary to associate each lexical item with as many descriptions as the number of different syntactic contexts in which the

* 180 Park Avenue, Florham Park, NJ 07932. E-mail: srini@research.att.com
† Department of Computer and Information Sciences and Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA 19104. E-mail: joshi@linc.cis.upenn.edu

lexical item can appear. This, of course, increases the local ambiguity for the parser. The parser has to decide which complex description out of the set of descriptions associated with each lexical item is to be used for a given reading of a sentence, even before combining the descriptions together. The obvious solution is to put the burden of this job entirely on the parser. The parser will eventually disambiguate all the descriptions and pick one per lexical item, for a given reading of the sentence. However, there is an alternate method of parsing that reduces the amount of disambiguation done by the parser. The idea is to locally check the constraints that are associated with the descriptions of lexical items to filter out incompatible descriptions.[1] During this disambiguation, the system can also exploit statistical information that can be associated with the descriptions based on their distribution in a corpus of parses.

We first employed these ideas in the context of Lexicalized Tree Adjoining grammars (LTAG) in Joshi and Srinivas (1994). Although presented with respect to LTAG, these techniques are applicable to other lexicalized grammars as well. In this paper, we present vastly improved supertag disambiguation results—from previously published 68% accuracy to 92% accuracy using a larger training corpus and better smoothing techniques. The layout of the paper is as follows: In Section 2, we present an overview of the robust parsing approaches. A brief introduction to Lexicalized Tree Adjoining grammars is presented in Section 3. Section 4 illustrates the goal of supertag disambiguation through an example. Various methods and their performance results for supertag disambiguation are discussed in detail in Section 5 and Section 6. In Section 7, we discuss the efficiency gained in performing supertag disambiguation before parsing. A robust and lightweight dependency analyzer that uses the supertag output is briefly presented in Section 8. In Section 9, we will discuss the applicability of supertag disambiguation to other lexicalized grammars.

2. Related Approaches

In recent years, there have been a number of attempts at robust parsing of natural language. They can be broadly categorized under two paradigms—finite-state-grammar-based parsers and statistical parsers. We briefly present these two paradigms and situate our approach to robust parsing relative to these paradigms.

2.1 Finite-State-Grammar-based Parsers

Finite-state-grammar-based approaches to parsing are exemplified by the parsing systems in Joshi, (1960), Abney (1990), Appelt et al. (1993), Roche (1993), Grishman (1995), Hobbs et al. (1997), Joshi and Hopely (1997), and Karttunen et al. (1997). These systems use grammars that are represented as cascaded finite-state regular expression recognizers. The regular expressions are usually hand-crafted. Each recognizer in the cascade provides a locally optimal output. The output of these systems is mostly in the form of noun groups and verb groups rather than constituent structure, often called a **shallow parse**. There are no clause-level attachments or modifier attachments in the shallow parse. These parsers always produce one output, since they use the longest-match heuristic to resolve cases of ambiguity when more than one regular expression

1 The use of descriptions for primitives to capture constraints locally has a precursor in AI. The Waltz algorithm (Waltz 1975) for labeling vertices of polygonal solid objects can be thought of in these terms. Waltz made the description of vertices more complex by including information about the incident edges, associated surfaces and other information. This increases the local ambiguity but the local constraints on the complex descriptions are strong enough to efficiently disambiguate the descriptions. Of course, Waltz did not use statistical information for disambiguation. See also Joshi (1998).

matches the input string at a given position. At present none of these systems use any statistical information to resolve ambiguity. The grammar itself can be partitioned into domain-independent and domain-specific regular expressions, which implies that porting to a new domain would involve rewriting the domain-dependent expressions. This approach has proved to be quite successful as a preprocessor in information extraction systems (Hobbs et al. 1995; Grishman 1995).

2.2 Statistical Parsers

Pioneered by the IBM natural language group (Fujisaki et al. 1989) and later pursued by, for example, Schabes, Roth, and Osborne (1993), Jelinek et al. (1994), Magerman (1995), Collins (1996), and Charniak (1997), this approach decouples the issue of well-formedness of an input string from the problem of assigning a structure to it. These systems attempt to assign some structure to every input string. The rules to assign a structure to an input are extracted automatically from hand-annotated parses of large corpora, which are then subjected to smoothing to obtain reasonable coverage of the language. The resultant set of rules are not linguistically transparent and are not easily modifiable. Lexical and structural ambiguity is resolved using probability information that is encoded in the rules. This allows the system to assign the most-likely structure to each input. The output of these systems consists of constituent analysis, the degree of detail of which is dependent on the detail of annotation present in the treebank that is used to train the system.

There are also parsers that use probabilistic (weighting) information in conjunction with hand-crafted grammars, for example, Black et al. (1993), Nagao (1994), Alshawi and Carter (1994), and Srinivas, Doran, and Kulick (1995). In these cases the probabilistic information is primarily used to rank the parses produced by the parser and not so much for the purpose of robustness of the system.

3. Lexicalized Grammars

Lexicalized grammars are particularly well-suited for the specification of natural language grammars. The lexicon plays a central role in linguistic formalisms such as LFG (Kaplan and Bresnan 1983), GPSG (Gazdar et al. 1985), HPSG (Pollard and Sag 1987), CCG (Steedman 1987), Lexicon Grammar (Gross 1984), LTAG (Schabes and Joshi 1991), Link Grammar (Sleator and Temperley 1991), and some version of GB (Chomsky 1992). Parsing, lexical semantics, and machine translation, to name a few areas, have all benefited from lexicalization. Lexicalization provides a clean interface for combining the syntactic and semantic information in the lexicon. We discuss the merits of lexicalization and other related issues in the context of partial parsing and briefly discuss Feature-based Lexicalized Tree Adjoining Grammars (LTAGs) as a representative of the class of lexicalized grammars.

Feature-based Lexicalized Tree Adjoining Grammar (FB-LTAG) (Joshi, Levy, and Takahashi 1975; Vijay-Shanker 1987; Schabes, Abeillé, and Joshi 1988; Vijay-Shanker and Joshi 1991; Joshi and Schabes 1996) is a tree-rewriting grammar formalism unlike context-free grammars and head grammars, which are string-rewriting formalisms. The primitive elements of FB-LTAGs are called **elementary trees**. Each elementary tree is associated with at least one lexical item on its frontier. The lexical item associated with an elementary tree is called the **anchor** of that tree. An elementary tree serves as a complex description of the anchor and provides a domain of locality over which the anchor can specify syntactic and semantic (predicate argument) constraints. Elementary trees are of two kinds: (a) **initial trees** and (b) **auxiliary trees**. In an FB-LTAG grammar for natural language, initial trees are phrase structure trees of simple sentences containing no recursion, while recursive structures are represented by auxiliary trees. Elementary trees are combined by substitution and adjunction operations. The result of combining the elementary trees is the **derived tree** and the process of combining the elementary trees to yield a parse of the sentence is represented by the **derivation tree**. The derivation tree can also be interpreted as a **dependency tree** with unlabeled arcs between words of the sentence. A more detailed discussion of LTAGs with an example and some of the key properties of elementary trees is presented in Appendix A.

4. Supertags

Part-of-speech disambiguation techniques (POS taggers) (Church 1988; Weischedel et al. 1993; Brill 1993) are often used prior to parsing to eliminate (or substantially reduce) the part-of-speech ambiguity. The POS taggers are all local in the sense that they use information from a limited context in deciding which tag(s) to choose for each word. As is well known, these taggers are quite successful.

In a lexicalized grammar such as the Lexicalized Tree Adjoining Grammar (LTAG), each lexical item is associated with at least one elementary structure (tree). The elementary structures of LTAG localize dependencies, including long-distance dependencies, by requiring that all and only the dependent elements be present within the same structure. As a result of this localization, a lexical item may be (and, in general, almost always is) associated with more than one elementary structure. We will call these elementary structures **supertags**, in order to distinguish them from the standard part-of-speech tags. Note that even when a word has a unique standard part of speech, say a verb (V), there will usually be more than one supertag associated with this word. Since there is only one supertag for each word (assuming there is no global ambiguity) when the parse is complete, an LTAG parser (Schabes, Abeillé, and Joshi 1988) needs to search a large space of supertags to select the right one for each word before combining them for the parse of a sentence. It is this problem of supertag disambiguation that we address in this paper.

Since LTAGs are lexicalized, we are presented with a novel opportunity to eliminate or substantially reduce the supertag assignment ambiguity by using local information, such as local lexical dependencies, prior to parsing. As in standard part-of-speech disambiguation, we can use local statistical information in the form of n-gram models based on the distribution of supertags in an LTAG parsed corpus. Moreover, since the supertags encode dependency information, we can also use information about the distribution of distances between a given supertag and its dependent supertags.

Note that as in standard part-of-speech disambiguation, supertag disambiguation could have been done by a parser. However, carrying out part-of-speech disambiguation prior to parsing makes the job of the parser much easier and therefore speeds it up. Supertag disambiguation reduces the work of the parser even further. After supertag disambiguation, we would have effectively completed the parse and the parser need "only" combine the individual structures; hence the term "almost parsing." This method can also be used to associate a structure to sentence fragments and in cases where the supertag sequence after disambiguation may not combine into a single structure.

4.1 Example of Supertagging

LTAGs, by virtue of possessing the Extended Domain of Locality (EDL) property,[2] associate with each lexical item, one elementary tree for each syntactic environment that

2 EDL is described in Appendix B.

Table 1
Examples of syntactic environments where the supertags shown in Figure 1 would be used.

Supertag	Construction	Example
α_1	Nominal Predicative	this is the *purchase*
α_2	Noun Phrase	the *price*
α_3	Topicalization	Almost everything, the price *includes*
α_4	Adjectival Predicative	this is *ancillary*
α_5	Noun Phrase	the *company*
β_1	Determiner	*the* company
β_2	Nominal Modifier	*purchase* order
α_6	Nominal Predicative Subject Extraction	what is the *price*
α_7	Imperative	*include* the share price
β_3	Determiner	*two* hundred men
β_4	Adjectival Modifier	*ancillary* unit
α_8	Nominal Predicative Subject Extraction	which are the *companies*
α_9	Noun Phrase	*purchases* have not increased.
α_{10}	Nominal Predicative	this is the *price*
α_{11}	Transitive Verb	the price *includes* everything
α_{12}	Adjectival Predicative Subject Extraction	what is *ancillary*
α_{13}	Noun Phrase	*companies* have not been profitable

the lexical item may appear in. As a result, each lexical item is invariably associated with more than one elementary tree. We call the elementary structures associated with each lexical item super parts-of-speech (super POS) or supertags.[3] Figure 1 illustrates a few elementary trees associated with each word of the sentence: *the purchase price includes two ancillary companies*. Table 1 provides an example context in which each supertag shown in Figure 1 would be used.

The example in Figure 2 illustrates the initial set of supertags assigned to each word of the sentence: *the purchase price includes two ancillary companies*. The order of the supertags for each lexical item in the example is not relevant. Figure 2 also shows the final supertag sequence assigned by the supertagger, which picks the best supertag sequence using statistical information (described in Section 6) about individual supertags and their dependencies on other supertags. The chosen supertags are combined to derive a parse. Without the supertagger, the parser would have to process combinations of the entire set of trees (at least the 17 trees shown); with it the parser need only process combinations of 7 trees.

5. Reducing Supertag Ambiguity Using Structural Information

The structure of the supertag can be best seen as providing admissibility constraints on syntactic environments in which it may be used. Some of these constraints can be checked locally. The following are a few constraints that can be used to determine the admissibility of a syntactic environment for a supertag:[4]

3 For the purpose of this paper, we suppress the features associated with the supertags.
4 Mitch Marcus pointed out that these tests are similar to the generalized shaper tests used in the Harvard Predictive Analyzer (Kuno 1966).

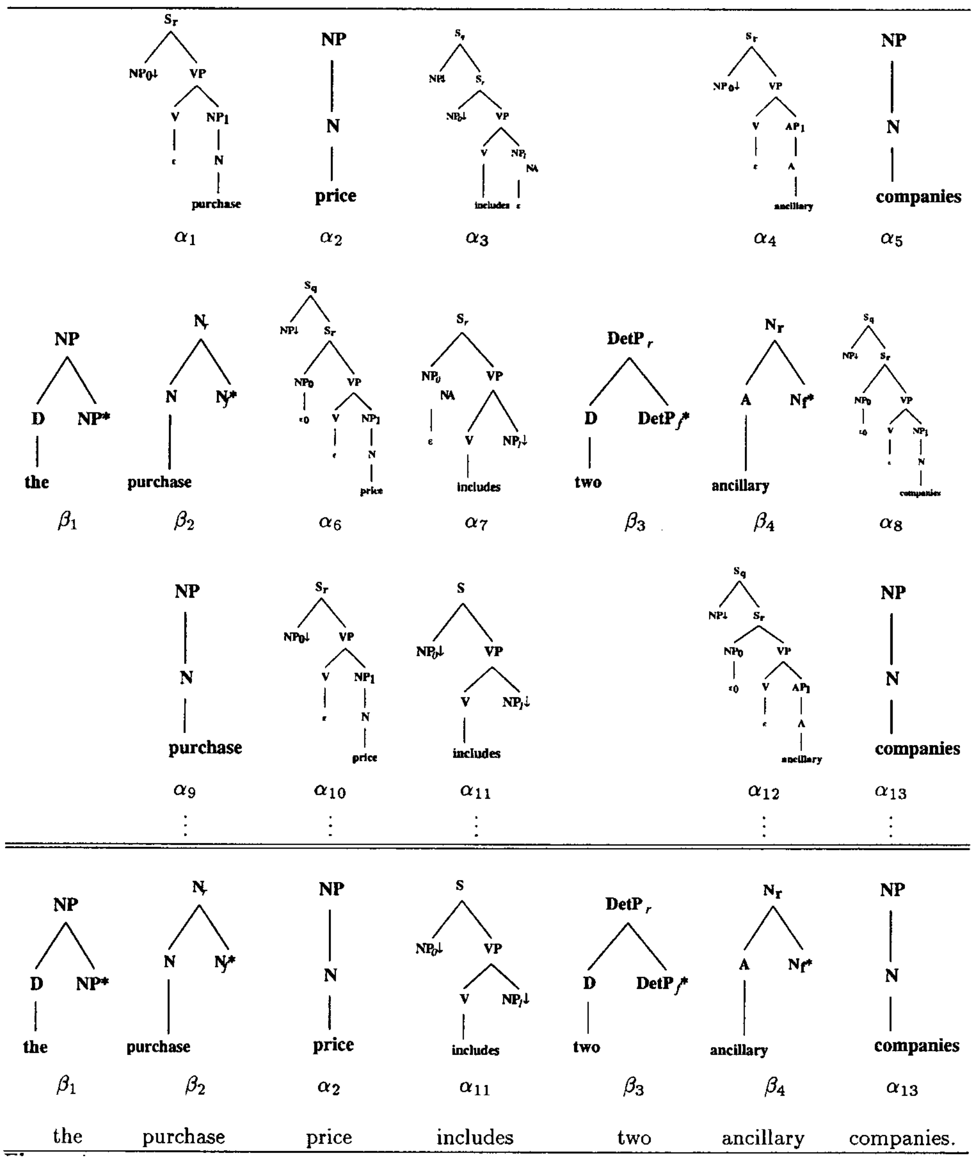

Figure 1
A selection of the supertags associated with each word of the sentence: the purchase price includes two ancillary companies.

- Span of the supertag: Span of a supertag is the minimum number of lexical items that the supertag can cover. Each substitution site of a supertag will cover at least one lexical item in the input. A simple rule can be used to eliminate supertags based on the **span constraint**: if the span of a supertag is larger than the input string, then the supertag cannot be used in any parse of the input string.

Sent:	the	purchase	price	includes	two	ancillary	companies.
Initial		α_1	α_2	α_3		α_4	α_5
Assignment	β_1	β_2	α_6	α_7	β_3	β_4	α_8
		α_9	α_{10}	α_{11}		α_{12}	α_{13}
		$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
Final Assignment	β_1	β_2	α_2	α_{11}	β_3	β_4	α_{13}

Figure 2
Supertag disambiguation for the sentence: the purchase price includes two ancillary companies.

Table 2
Supertag ambiguity with and without the use of structural constraints.

System	Total # of Words	Average # of Supertags/Word
Without Structural Constraints	48,783	47.0
With Structural Constraints	48,783	25.0

- Left (Right) span constraint: If the span of the supertag to the left (right) of the anchor is larger than the length of the string to the left (right) of the word that anchors the supertag, then the supertag cannot be used in any parse of the input string.
- Lexical items in the supertag: A supertag can be eliminated if the terminals appearing on the frontier of the supertag do not appear in the input string.

Supertags with the built-in lexical item *by*, that represent passive constructions are typically eliminated from being considered during the parse of an active sentence.

More generally, these constraints can be used to eliminate supertags that cannot have their features satisfied in the context of the input string. An example of this is the elimination of supertag that requires a *wh*+ NP when the input string does not contain *wh*-words.

Table 2 indicates the decrease in supertag ambiguity for 2,012 WSJ sentences (48,763 words) by using the structural constraints relative to the supertag ambiguity without the structural constraints.[5]

These filters prove to be very effective in reducing supertag ambiguity. The graph in Figure 3 plots the number of supertags at the sentence level for sentences of length 2 to 50 words with and without the filters. As can be seen from the graph, the supertag ambiguity is significantly lower when the filters are used. The graph in Figure 4 shows the percentage drop in supertag ambiguity due to filtering for sentences of length 2 to 50 words. As can be seen, the average reduction in supertag ambiguity is about 50%. This means that given a sentence, close to 50% of the supertags can be eliminated even before parsing begins by just using structural constraints of the supertags. This reduction in supertag ambiguity speeds up the parser significantly. In fact, the supertag

5 WSJ Section 20 of the Penn Treebank.

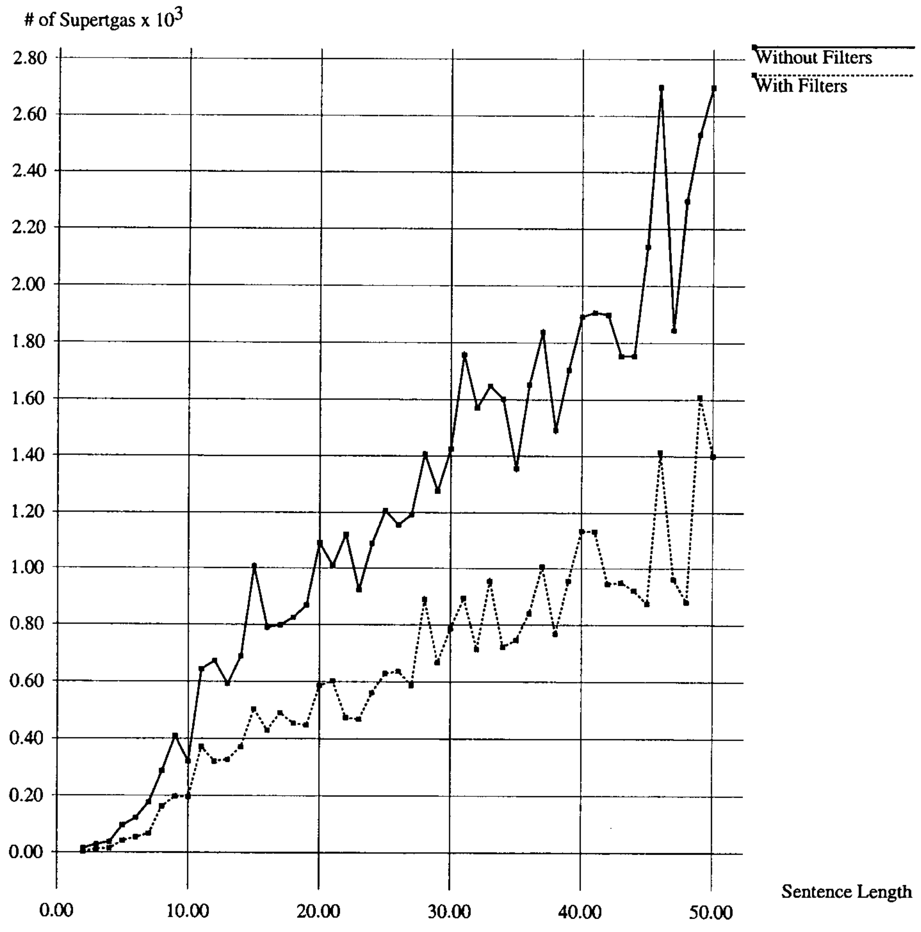

Figure 3
Comparison of number of supertags with and without filtering for sentences of length 2 to 50 words.

ambiguity in XTAG system is so large that the parser is prohibitively slow without the use of these filters.

Table 3 tabulates the reduction of supertag ambiguity due to the filters against various parts of speech.[6] Verbs in all their forms contribute most to the problem of supertag ambiguity and most of the supertag ambiguity for verbs is due to light verbs and verb particles. The filters are very effective in eliminating over 50% of the verb anchored supertags.

Even though structural constraints are effective in reducing supertag ambiguity, the search space for the parser is still sufficiently large. In the next few sections, we present stochastic and rule-based approaches to supertag disambiguation.

6 The description of the part-of-speech tags is provided in Marcus, Santorini, and Marcinkiewicz (1993).

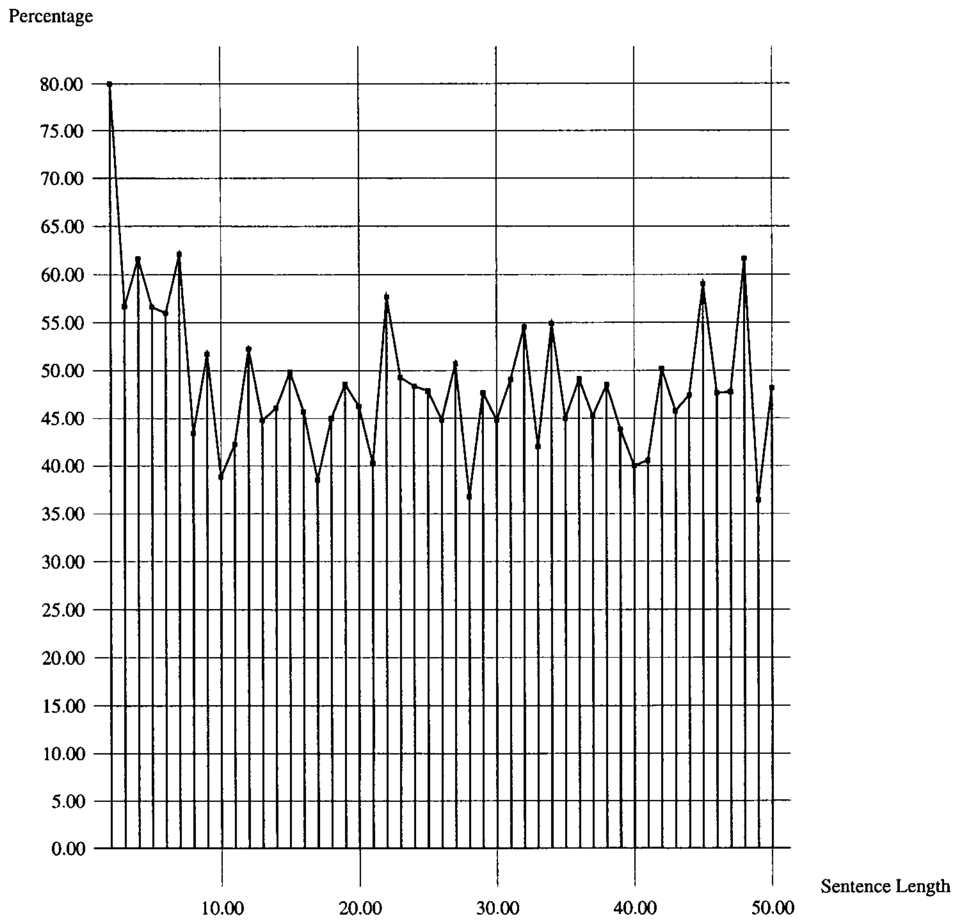

Figure 4
Percentage drop in the number of supertags with and without filtering for sentences of length 2 to 50 words.

6. Models, Data, Experiments, and Results

Before proceeding to discuss the various models for supertag disambiguation, we would like to trace the time course of development of this work. We do this not only to show the improvements made to the early work reported in our 1994 paper (Joshi and Srinivas 1994), but also to explain the rationale for choosing certain models of supertag disambiguation over others. We summarize the early work in the following subsection.

6.1 Early Work

As reported in Joshi and Srinivas (1994), we experimented with a trigram model as well as the dependency model for supertag disambiguation. The trigram model that was trained on (part-of-speech, supertag) pairs, instead of (words, supertag) pairs, collected from the LTAG derivations of 5,000 WSJ sentences and tested on 100 WSJ sentences produced a correct supertag for 68% of the words in the test set. We have since significantly improved the performance of the trigram model by using a larger

Table 3
The effect of filters on supertag ambiguity tabulated against part of speech.

POS	Average # of Supertags without Filters	Average # of Supertags with Filters	Percentage Drop in Supertag Ambiguity
VBP	516.5	250.0	51.6
VB	435.8	224.9	48.4
VBD	209.0	100.7	51.8
VBN	188.2	74.7	60.3
MD	167.2	121.0	27.6
VBZ	165.1	71.6	56.6
VBG	100.7	49.8	50.5
RP	34.5	30.9	10.5
IN	24.3	20.9	14.0
JJS	23.8	12.7	46.9
WRB	23.1	14.3	38.2
JJR	22.7	14.2	37.7
JJ	21.7	13.5	37.9
,	20.0	10.7	46.6
NN	19.8	10.7	46.0
NNS	17.0	10.5	38.6
NNP	15.0	10.2	31.9
NNPS	15.0	10.2	32.1
LS	15.0	15.0	0.0
FW	15.0	15.0	0.0
-RRB-	15.0	10.7	28.4
-LRB-	15.0	12.3	18.0
RBR	14.9	9.5	36.3
RBS	14.9	6.1	59.2
CC	14.8	3.4	76.9
EX	14.0	5.8	58.7
CD	13.3	9.9	25.8
TO	11.3	10.8	4.5
PRP	10.7	5.3	50.2
UH	10.0	3.0	70.0
RB	10.0	5.3	46.4
"	6.0	3.2	46.7
:	5.5	3.2	42.1
PDT	5.4	4.9	9.0
WP	4.6	2.9	35.8
WP$	4.0	1.8	56.2
DT	3.9	3.1	21.8
PRP$	3.8	2.9	22.2
.	3.0	1.0	65.4
POS	2.5	2.1	13.9
WDT	1.2	1.1	5.5

training set and incorporating smoothing techniques. We present a detailed discussion of the model and its performance on a range of corpora in Section 6.5. In Section 6.2, we briefly mention the dependency model of supertagging that was reported in the earlier work.

6.2 Dependency Model

In an *n*-gram model for disambiguating supertags, dependencies between supertags that appear beyond the *n*-word window cannot be incorporated. This limitation can be overcome if no a priori bound is set on the size of the window but instead a probability distribution of the distances of the dependent supertags for each supertag is maintained. We define dependency between supertags in the obvious way: A supertag is dependent on another supertag if the former substitutes or adjoins into the latter. Thus, the substitution and the foot nodes of a supertag can be seen as specifying dependency requirements of the supertag. The probability with which a supertag depends on another supertag is collected from a corpus of sentences annotated with derivation structures. Given a set of supertags for each word and the dependency information between pairs of supertags, the objective of the dependency model is to compute the most likely dependency linkage that spans the entire string. The result of producing the dependency linkage is a sequence of supertags, one for each word of the sentence along with the dependency information.

Since first reported in Joshi and Srinivas (1994), we have not continued experiments using this model of supertagging, primarily for two reasons. We are restrained by the lack of a large corpus of LTAG parsed derivation structures that is needed to reliably estimate the various parameters of this model. We are currently in the process of collecting a large LTAG parsed WSJ corpus, with each sentence annotated with the correct derivation. A second reason for the disuse of the dependency model for supertagging is that the objective of supertagging is to see how far local techniques can be used to disambiguate supertags even before parsing begins. The dependency model, in contrast, is too much like full parsing and is contrary to the spirit of supertagging.

6.3 N-gram Models with Smoothing

We have improved the performance of the trigram model by incorporating smoothing techniques into the model and training the model on a larger training corpus. We have also proposed some new models for supertag disambiguation. In this section, we discuss these developments in detail.

Two sets of data are used for training and testing the models for supertag disambiguation. The first set has been collected by parsing the Wall Street Journal[7], IBM Manual, and ATIS corpora using the wide-coverage English grammar being developed as part of the XTAG system (Doran et al. 1994). The correct derivation from all the derivations produced by the XTAG system was picked for each sentence from these corpora.

The second and larger data set was collected by converting the Penn Treebank parses of the Wall Street Journal sentences. The objective was to associate each lexical item of a sentence with a supertag, given the phrase structure parse of the sentence. This process involved a number of heuristics based on local tree contexts. The heuristics made use of information about the labels of a word's dominating nodes (parent, grandparent, and great-grandparent), labels of its siblings (left and right) and siblings of its parent. An example of the result of this conversion is shown in Figure 5. It must be noted that this conversion is not perfect and is correct only to a first order of approximation owing mostly to errors in conversion and lack of certain kinds of information such as distinction between adjunct and argument preposition phrases, in the Penn Treebank parses. Even though the converted supertag corpus can be refined further, the corpus in its present form has proved to be an invaluable resource in improving the performance of the supertag models as is discussed in the following sections.

7 Sentences of length $\leq$ 15 words.

```
( ("S"
   ("NP-SBJ" ("NNP" "Mr.") ("NNP" "Vinken") )
   ("VP" ("VBZ" "is")
    ("NP-PRD"
     ("NP" ("NN" "chairman") )
     ("PP" ("IN" "of")
      ("NP"
       ("NP" ("NNP" "Elsevier") ("NNP" "N.V.") )
       ("," ",")
       ("NP" ("DT" "the") ("NNP" "Dutch") ("VBG" "publishing") ("NN" "group
") )))))
   ("." ".") ))
```

Mr.//NNP//B_Nn	(noun modifier)
Vinken//NNP//A_NXN	(head noun)
is//VBZ//B_Vvx	(auxiliary verb)
chairman//NN//A_nxON1	(predicative noun)
of//IN//B_nxPnx	(noun-attached preposition)
Elsevier//NNP//B_Nn	(noun modifier)
N.V.//NNP//A_NXN	(head noun)
,//,//B_nxPUnxpu	(appositive comma)
the//DT//B_Dnx	(determiner)
Dutch//NNP//B_Nn	(noun modifier)
publishing//VBG//B_Vn	(participle verb, nominal modifier)
group//NN//A_NXN	(head noun)
.//.//B_sPU	(sentence punctuation)

Figure 5
The phrase structure tree and the supertags obtained from the phrase structure tree for the WSJ sentence: *Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.*

6.4 Unigram Model

Using structural information to filter out supertags that cannot be used in any parse of the input string reduces the supertag ambiguity but obviously does not eliminate it completely. One method of disambiguating the supertags assigned to each word is to order the supertags by the lexical preference that the word has for them. The frequency with which a certain supertag is associated with a word is a direct measure of its lexical preference for that supertag. Associating frequencies with the supertags and using them to associate a particular supertag with a word is clearly the simplest means of disambiguating supertags. Therefore a unigram model is given by:

$$Supertag(w_i) = t_k \ni argmax_{t_k} Pr(t_k \mid w_i). \tag{1}$$

where

$$Pr(t_k \mid w_i) = \frac{frequency(t_k, w_i)}{frequency(w_i)} \tag{2}$$

Thus, the most frequent supertag that a word is associated with in a training corpus is selected as the supertag for the word according to the unigram model. For the words that do not appear in the training corpus we back off to the part of speech of the word and use the most frequent supertag associated with that part of speech as the supertag for the word.

Table 4
Results from the unigram supertag model.

Data Set	Training Set	Test Set	Top n Supertags	% Success
XTAG Parses	8,000	3,000	$n = 1$	73.4%
			$n = 2$	80.2%
			$n = 3$	80.8%
Converted Penn Treebank Parses	1,000,000	47,000	$n = 1$	77.2%
			$n = 2$	87.0%
			$n = 3$	91.5%

6.4.1 Experiments and Results. We tested the performance of the unigram model on the previously discussed two sets of data. The words are first assigned standard parts of speech using a conventional tagger (Church 1988) and then are assigned supertags according to the unigram model. A word in a sentence is considered correctly supertagged if it is assigned the same supertag as it is associated with in the correct parse of the sentence. The results of these experiments are tabulated in Table 4.

Although the performance of the unigram model for supertagging is significantly lower than the performance of the unigram model for part-of-speech tagging (91% accuracy), it performed much better than expected considering the size of the supertag set is much larger than the size of part-of-speech tag set. One of the reasons for this high performance is that the most frequent supertag for the most frequent words—determiners, nouns, and auxiliary verbs—is the correct supertag most of the time. Also, backing off to the part of speech helps in supertagging unknown words, which most often are nouns. The bulk of the errors committed by the unigram model is incorrectly tagged verbs (subcategorization and transformation), prepositions (noun attached vs. verb attached) and nouns (head vs. modifier noun).

6.5 N-gram Model

We first explored the use of trigram model of supertag disambiguation in Joshi and Srinivas (1994). The trigram model was trained on (part-of-speech, supertag) pairs collected from the LTAG derivations of 5,000 WSJ sentences and tested on 100 WSJ sentences. It produced a correct supertag for 68% of the words in the test set. A major drawback of this early work was that it used no lexical information in the supertagging process as the training material consisted of (part-of-speech, supertag) pairs. Since that early work, we have improved the performance of the model by incorporating lexical information and sophisticated smoothing techniques, as well as training on larger training sets. In this section, we present the details and the performance evaluation of this model.

In a unigram model, a word is always associated with the supertag that is most preferred by the word, irrespective of the context in which the word appears. An alternate method that is sensitive to context is the n-gram model. The n-gram model takes into account the contextual dependency probabilities between supertags within a window of n words in associating supertags to words. Thus, the most probable supertag sequence for an n-word sentence is given by:

$$\hat{T} = argmax_T Pr(T_1, T_2, \ldots, T_N) * Pr(W_1, W_2, \ldots, W_N \mid T_1, T_2, \ldots, T_N) \qquad (3)$$

where T_i is the supertag for word W_i.

To compute this using only local information, we approximate, assuming that the probability of a word depends only on its supertag

$$Pr(W_1, W_2, \ldots, W_N \mid T_1, T_2, \ldots, T_N) \approx \prod_{i=1}^{N} Pr(W_i \mid T_i) \tag{4}$$

and also use an n-gram (trigram, in this case) approximation

$$Pr(T_1, T_2, \ldots, T_N) \approx \prod_{i=1}^{N} Pr(T_i \mid T_{i-2}, T_{i-1}) \tag{5}$$

The term $\Pr(T_i \mid T_{i-2}, T_{i-1})$ is known as the **contextual probability** since it indicates the size of the context used in the model and the term $\Pr(W_i \mid T_i)$ is called the **word emit probability** since it is the probability of emitting the word W_i given the tag T_i. These probabilities are estimated using a corpus where each word is tagged with its correct supertag.

The contextual probabilities were estimated using the relative frequency estimates of the contexts in the training corpus. To estimate the probabilities for contexts that do not appear in the training corpus, we used the Good-Turing discounting technique (Good 1953) combined with Katz's back off model (Katz 1987). The idea here is to discount the frequencies of events that occur in the corpus by an amount related to their frequencies and utilize this discounted probability mass in the back off model to distribute to unseen events. Thus, the Good-Turing discounting technique estimates the frequency of unseen events based on the distribution of the frequency of the counts of observed events in the corpus. If r is the observed frequency of an event, and N_r is the number of events with the observed frequency r, and N is the total number of events, then the probability of an unseen event is given by N_1/N. Furthermore, the frequencies of the observed events are adjusted so that the total probability of all events sums to one. The adjusted frequency for observed events, r^*, is computed as

$$r^* = (r+1) * \frac{N_{r+1}}{N_r} \tag{6}$$

Once the frequencies of the observed events are discounted and the frequencies for unseen events are estimated, Katz's back off model is used. In this technique, if the observed frequency of an $<$ n-gram, supertag$>$ sequence is zero then its probability is computed based on the observed frequency of an $(n-1)$-gram sequence. Thus,

$$\begin{aligned}
\tilde{Pr}(T_3|T_1, T_2) &= Pr(T_3|T_1, T_2) \ \mathit{if}\ Pr(T_3|T_1, T_2) > 0 \\
&= \alpha(T_1, T_2) * \tilde{Pr}(T_3|T_2) \quad \mathit{if}\ Pr(T_2|T_1) > 0 \\
&= Pr(T_3|T_2) \text{ otherwise}
\end{aligned}$$

$$\begin{aligned}
\tilde{Pr}(T_2|T_1) &= Pr(T_2|T_1) \quad \mathit{if}\ Pr(T_2|T_1) > 0 \\
&= \beta(T_1) * Pr_1(T_2) \text{ otherwise}
\end{aligned}$$

where $\alpha(T_i, T_j)$ and $\beta(T_k)$ are constants to ensure that the probabilities sum to one.

The word emit probability for the (word, supertag) pairs that appear in the training corpus is computed using the relative frequency estimates as shown in Equation 7. For the (word, supertag) pairs that do not appear in the corpus, the word emit probability is estimated as shown in Equation 8. Some of the word features used in our implementation include prefixes and suffixes of length less than or equal to three characters, capitalization, and digit features.

$$
\begin{aligned}
Pr(W_i|T_i) &= \frac{N(W_i, T_i)}{N(T_i)} \; if \; N(W_i, T_i) > 0 && (7)\\
&= Pr(UNK|T_i) * Pr(word\_features(W_i)|T_i) \text{ otherwise} && (8)
\end{aligned}
$$

The counts for the (word, supertag) pairs for the words that do not appear in the corpus is estimated using the leaving-one-out technique (Niesler and Woodland 1996; Ney, Essen, and Kneser 1995). A token *UNK* is associated with each supertag and its count N_{UNK} is estimated by:

$$
\begin{aligned}
Pr(UNK|T_j) &= \frac{N_1(T_j)}{N(T_j) + \eta}\\
N_{UNK}(T_j) &= \frac{Pr(UNK|T_j) * N(T_j)}{1 - Pr(UNK|T_j)}
\end{aligned}
$$

where $N_1(T_j)$ is the number of words that are associated with the supertag T_j that appear in the corpus exactly once. $N(T_j)$ is the frequency of the supertag T_j and $N_{UNK}(T_j)$ is the estimated count of UNK in T_j. The constant η is introduced so as to ensure that the probability is not greater than one, especially for supertags that are sparsely represented in the corpus.

We use word features similar to the ones used in Weischedel et al. (1993), such as capitalization, hyphenation, and endings of words, for estimating the word emit probability of unknown words.

6.5.1 Experiments and Results. We tested the performance of the trigram model on various domains such as the Wall Street Journal (WSJ), the IBM Manual corpus and the ATIS corpus. For the IBM Manual corpus and the ATIS domains, a supertag annotated corpus was collected using the parses of the XTAG system (Doran et al. 1994) and selecting the correct analysis for each sentence. The corpus was then randomly split into training and test material. Supertag performance is measured as the percentage of words that are correctly supertagged by a model when compared with the key for the words in the test corpus.

Experiment 1: (Performance on the Wall Street Journal corpus). We used the two sets of data, from the XTAG parses and from the conversion of the Penn Treebank parses to evaluate the performance of the trigram model. Table 5 shows the performance on the two sets of data. The first data set, data collected from the XTAG parses, was split into 8,000 words of training and 3,000 words of test material. The data collected from converting the Penn Treebank was used in two experiments differing in the size of the training corpus—200,000 words[8] and 1,000,000 words[9]—and tested on 47,000 words[10]. A total of 300 different supertags were used in these experiments.

Experiment 2: (Performance on the IBM Manual Corpus and ATIS). For testing the performance of the trigram supertagger on the IBM Manual corpus, a set of 14,000 words

8 Sentences in WSJ Sections 15 through 18 of Penn Treebank.
9 Sentences in WSJ Sections 00 through 24, except Section 20 of Penn Treebank.
10 Sentences in WSJ Section 20 of Penn Treebank.

Table 5
Performance of the supertagger on the WSJ corpus.

Data Set	Size of Training Set (Words)	Training	Size of Test Set (Words)	% Correct
XTAG Parses	8,000	Unigram (Baseline)	3,000	73.4%
		Trigram	3,000	86.0%
Converted Penn Treebank Parses	200,000	Unigram (Baseline)	47,000	75.3%
		Trigram	47,000	90.9%
	1,000,000	Unigram (Baseline)	47,000	77.2%
		Trigram	47,000	92.2%

Table 6
Performance of the supertagger on the IBM Manual corpus and ATIS corpus.

Corpus	Size of Training Set (Words)	Training	Size of Test Set (Words)	% Correct
IBM Manual	14,000	Unigram (Baseline)	1,000	77.8%
		Trigram	1,000	90.3%
ATIS	1,500	Unigram (Baseline)	400	85.7%
		Trigram	400	93.8%

correctly supertagged was used as the training corpus and a set of 1,000 words was used as a test corpus. The performance of the supertagger on this corpus is shown in Table 6. Performance on the ATIS corpus was evaluated using a set of 1,500 words correctly supertagged as the training corpus and a set of 400 words as a test corpus. The performance of the supertagger on the ATIS corpus is also shown in Table 6.

As expected, the performance on the ATIS corpus is higher than that of the WSJ and the IBM Manual corpus despite the extremely small training corpus. Also, the performance of the IBM Manual corpus is better than the WSJ corpus when the size of the training corpus is taken into account. The baseline for the ATIS domain is remarkably high due to the repetitive constructions and limited vocabulary in that domain. This is also true for the IBM Manual corpus, although to a lesser extent. The trigram model of supertagging is attractive for limited domains since it performs quite well with relatively insignificant amounts of training material. The performance of the supertagger can be improved in an iterative fashion by using the supertagger to supertag larger amounts of training material, which can be quickly hand-corrected and used to train a better-performing supertagger.

6.5.2 Effect of Lexical versus Contextual Information. Lexical information contributes most to the performance of a POS tagger, since the baseline performance of assigning the most likely POS for each word produces 91% accuracy (Brill 1993). Contextual information contributes relatively a small amount towards the performance, improving it from 91% to 96–97%, a 5.5% improvement. In contrast, contextual information has greater effect on the performance of the supertagger. As can be seen, from the above experiments, the baseline performance of the supertagger is about 77% and the performance improves to about 92% with the inclusion of contextual information, an improvement of 19.5%. The relatively low baseline performance for the supertagger is a direct consequence of the fact that there are many more supertags per word than there are POS tags. Further, since many combinations of supertags are not possible, contextual information has a larger effect on the performance of the supertagger.

6.6 Error-driven Transformation-based Tagger

In an error-driven transformation-based (EDTB) tagger (Brill 1993), a set of pattern-action templates that include predicates that test for features of words appearing in the context of interest are defined. These templates are then instantiated with the appropriate features to obtain transformation rules. The effectiveness of a transformation rule to correct an error and the relative order of application of the rules are learned using a corpus. The learning procedure takes a gold corpus in which the words have been correctly annotated and a training corpus that is derived from the gold corpus by removing the annotations. The objective in the learning phase is to learn the optimum ordering of rule applications so as to minimize the number of tag mismatches between the training and the reference corpus.

6.6.1 Experiments and Results. A EDTB model has been trained using templates defined on a three-word window. We trained the templates on 200,000 words[11] and tested on 47,000 words[12] of the WSJ corpus. The model performed at an accuracy of 90%. The EDTB model provides a great deal of flexibility to integrate domain-specific and linguistic information into the model. However, a major drawback of this approach is that the training procedure is extremely slow, which prevented us from training on the 1,000,000 word corpus.

7. Supertagging before Parsing

The output of the supertagger, an almost parse, has been used in a variety of applications including information retrieval (Chandrasekar and Srinivas 1997b, 1997c, 1997d) and information extraction (Doran et al. 1997), text simplification (Chandrasekar, Doran, and Srinivas 1996, Chandrasekar and Srinivas 1997a), and language modeling (Srinivas 1996) to illustrate that supertags provide an appropriate level of lexical description needed for most applications.

The output of the supertagger has also been used as a front end to a lexicalized grammar parser. As mentioned earlier, a lexicalized grammar parser can be conceptualized to consist of two stages (Schabes, Abeillé, and Joshi 1988). In the first stage, the parser looks up the lexicon and selects all the supertags associated with each word of the sentence to be parsed. In the second stage, the parser searches the lattice of selected supertags in an attempt to combine them using substitution and adjunction operations so as to yield a derivation that spans the input string. At the end of the second stage, the parser would not only have parsed the input, but would have associated a small set of (usually one) supertags with each word.

The supertagger can be used as a front end to a lexicalized grammar parser so as to prune the search-space of the parser even before parsing begins. It should be clear that by reducing the number of supertags that are selected in the first stage, the search-space for the second stage can be reduced significantly and hence the parser can be made more efficient. Supertag disambiguation techniques, as discussed in the

11 WSJ Sections 15 to 18 of the Penn Treebank.
12 WSJ Section 20 of the Penn Treebank.

Table 7
Performance improvement of 3-best supertagger over the 1-best supertagger on the WSJ corpus.

Data Set	Size of Test Set (Words)	Size of Training Set (Words)	Training	% Correct
Converted Penn Treebank Parses	47,000	200,000	Trigram (Best Supertag)	90.9%
			Trigram (3-Best Supertags)	95.8%
		1,000,000	Trigram (Best Supertag)	92.2%
			Trigram (3-Best Supertags)	97.1%

previous sections, attempt to disambiguate the supertags selected in the first pass, based on lexical preferences and local lexical dependencies, so as to ideally select one supertag for each word. Once the supertagger selects the appropriate supertag for each word, the second stage of the parser is needed only to combine the individual supertags to arrive at the parse of the input. Tested on about 1,300 WSJ sentences with each word in the sentence correctly supertagged, the LTAG parser took approximately 4 seconds per sentence to yield a parse (combine the supertags and perform feature unification). In contrast, the same 1,300 WSJ sentences without the supertag annotation took nearly 120 seconds per sentence to yield a parse. Thus the parsing speedup gained by this integration is a factor of about 30.

In the XTAG system, we have integrated the trigram supertagger as a front end to an LTAG parser to pick the appropriate supertag for each word even before parsing begins. However, a drawback of this approach is that the parser would fail completely if any word of the input is incorrectly tagged by the supertagger. This problem could be circumvented to an extent by extending the supertagger to produce *n*-best supertags for each word. Although this extension would increase the load on the parser, it would certainly improve the chances of arriving at a parse for a sentence. In fact, Table 7 presents the performance of the supertagger that selects, at most, the top three supertags for each word. The optimum number of supertags to output to balance the success rate of the parser against the efficiency of the parser must be determined empirically.

A more serious limitation of this approach is that it fails to parse ill-formed and extragrammatical strings such as those encountered in spoken utterances and unrestricted texts. This is due to the fact that the Earley-style LTAG parser attempts to combine the supertags to construct a parse that spans the entire string. In cases where the supertag sequence for a string cannot be combined into a unified structure, the parser fails completely. One possible extension to account for ill-formed and extragrammatical strings is to extend the Earley parser to produce partial parses for the fragments whose supertags can be combined. An alternate method of computing dependency linkages robustly is presented in the next section.

8. Lightweight Dependency Analyzer

Supertagging associates each word with a unique supertag. To establish the dependency links among the words of the sentence, we exploit the dependency requirements encoded in the supertags. Substitution nodes and foot nodes in supertags serve as slots that must be filled by the arguments of the anchor of the supertag. A substitution slot of a supertag is filled by the complements of the anchor while the foot node of a supertag is filled by a word that is being modified by the supertag. These argument slots have a polarity value reflecting their orientation with respect to the anchor of the supertag. Also associated with a supertag is a list of internal nodes (including the root node) that appear in the supertag. Using the structural information coupled with the argument requirements of a supertag, a simple heuristic-based, linear time, deterministic algorithm (which we call a lightweight dependency analyzer (LDA)) produces dependency linkages not necessarily spanning the entire sentence. The LDA can produce a number of partial linkages, since it is driven primarily by the need to satisfy local constraints without being driven to construct a single dependency linkage that spans the entire input. This, in fact, contributes to the robustness of LDA and promises to be a useful tool for parsing sentence fragments that are rampant in speech utterances, as exemplified by the Switchboard corpus.

Tested on section 20 of the Wall Street Journal corpus, which contained 47,333 dependency links in the gold standard, the LDA, trained on 200,000 words, produced 38,480 dependency links correctly, resulting in a recall score of 82.3%. Also, a total of 41,009 dependency links were produced by the LDA, resulting in a precision score of 93.8%. A detailed evaluation of the LDA is presented in Srinivas (1997b).

9. Applicability of Supertagging to other Lexicalized Grammars

Although we have presented supertagging in the context of LTAG, it is applicable to other lexicalized grammar formalisms such as CCG (Steedman 1997), HPSG (Pollard and Sag 1987), and LFG (Kaplan and Bresnan 1983). We have implemented a broad coverage CCG grammar (Doran and Srinivas 1994) containing about 80 categories based on the XTAG English grammar. These categories have been used to tag the same training and test corpora used in the supertagging experiments discussed in this paper and a supertagger to disambiguate the CCG categories has been developed. We are presently analyzing the performance of the supertagger using the LTAG trees and the CCG categories.

The idea of supertagging can also be applied to a grammar in HPSG formalism indirectly, by compiling the HPSG grammar into an LTAG grammar (Kasper et al. 1995). A more direct approach would be to tag words with feature structures that represent supertags (Kempe 1994). For LFG, the lexicalized subset of fragments used in the LFG-DOP model (Bod and Kaplan 1998) can be seen as supertags.

An approach that is closely related to supertagging is the reductionist approach to parsing that is being carried out under the Constraint Grammar framework (Karlsson et al. 1994; Voutilainen 1994; Tapanainen and Järvinen 1994). In this framework, each word is associated with the set of possible functional tags that it may be assigned in the language. This constitutes the lexicon. The grammar consists of a set of rules that eliminate functional tags for words based on the context of a sentence. Parsing a sentence in this framework amounts to eliminating as many implausible functional tags as possible for each word, given the context of the sentence. The resultant output structure might contain significant syntactic ambiguity, which may not have been eliminated by the rule applications, thus producing almost parses. Thus, the reductionist approach to parsing is similar to supertagging in that both view parsing as tagging with rich descriptions. However, the key difference is that the tagging is done in a probabilistic setting in the supertagging approach while it is rule based in the constraint grammar approach.

We are currently developing supertaggers for other languages. In collaboration with Anne Abeillé and Marie-Helene Candito of the University of Paris, using their French TAG grammar, we have developed a supertagger for French. We are currently working on evaluating the performance of this supertagger. Also, the annotated corpora necessary for training supertaggers for Korean and Chinese are under development at the University of Pennsylvania.

A version of the supertagger trained on the WSJ corpus is available under GNU Public License from http://www.cis.upenn.edu/~xtag/swrelease.html.

10. Conclusions

In this paper, we have presented a novel approach to robust parsing distinguished from the previous approaches to robust parsing by integrating the flexibility of linguistically motivated lexical descriptions with the robustness of statistical techniques. By associating rich descriptions (supertags) that impose complex constraints in a local context, we have been able to use local computational models for effective supertag disambiguation. A trigram supertag disambiguation model, trained on 1,000,000 (word, supertag) pairs of the Wall Street Journal corpus, performs at an accuracy level of 92.2%. After disambiguation, we have effectively completed the parse of the sentence, creating an almost parse, in that the parser need only combine the selected structures to arrive at a parse for the sentence. We have presented a lightweight dependency analyzer (LDA) that takes the output of the supertagger and uses the dependency requirements of the supertags to produce a dependency linkage for a sentence. This method can also serve to parse sentence fragments in cases where the supertag sequence after disambiguation may not combine to form a single structure. This approach is applicable to all lexicalized grammar parsers.

Appendix A: Feature-based Lexicalized Tree Adjoining Grammar

Feature-based Lexicalized Tree Adjoining Grammar (FB-LTAG) is a tree-rewriting grammar formalism, unlike context-free Grammars and head grammars, which are string-rewriting formalisms. FB-LTAGs trace their lineage to Tree Adjunct Grammars (TAGs), which were first developed in Joshi, Levy, and Takahashi (1975) and later extended to include unification-based feature structures (Vijay-Shanker 1987; Vijay-Shanker and Joshi 1991) and lexicalization (Schabes, Abeillé, and Joshi 1988). For a more recent and comprehensive reference, see Joshi and Schabes (1996).

The primitive elements of FB-LTAGs are called elementary trees. Each elementary tree is associated with at least one lexical item on its frontier. The lexical item associated with an elementary tree is called the anchor of that tree. An elementary tree serves as a complex description of the anchor and provides a domain of locality over which the anchor can specify syntactic and semantic (predicate argument) constraints. Elementary trees are of two kinds: (a) Initial Trees and (b) Auxiliary Trees. In an FB-LTAG grammar for natural language, initial trees are phrase structure trees of simple sentences containing no recursion, while recursive structures are represented by auxiliary trees.

Examples of initial trees (αs) and auxiliary trees (βs) are shown in Figure 6. Nodes on the frontier of initial trees are marked as substitution sites by a "$\downarrow$", while exactly one node on the frontier of an auxiliary tree, whose label matches the label of the root of the tree, is marked as a foot node by a "$*$". The other nodes on the frontier of an auxiliary tree are marked as substitution sites.

Each node of an elementary tree is associated with two feature structures (FS),

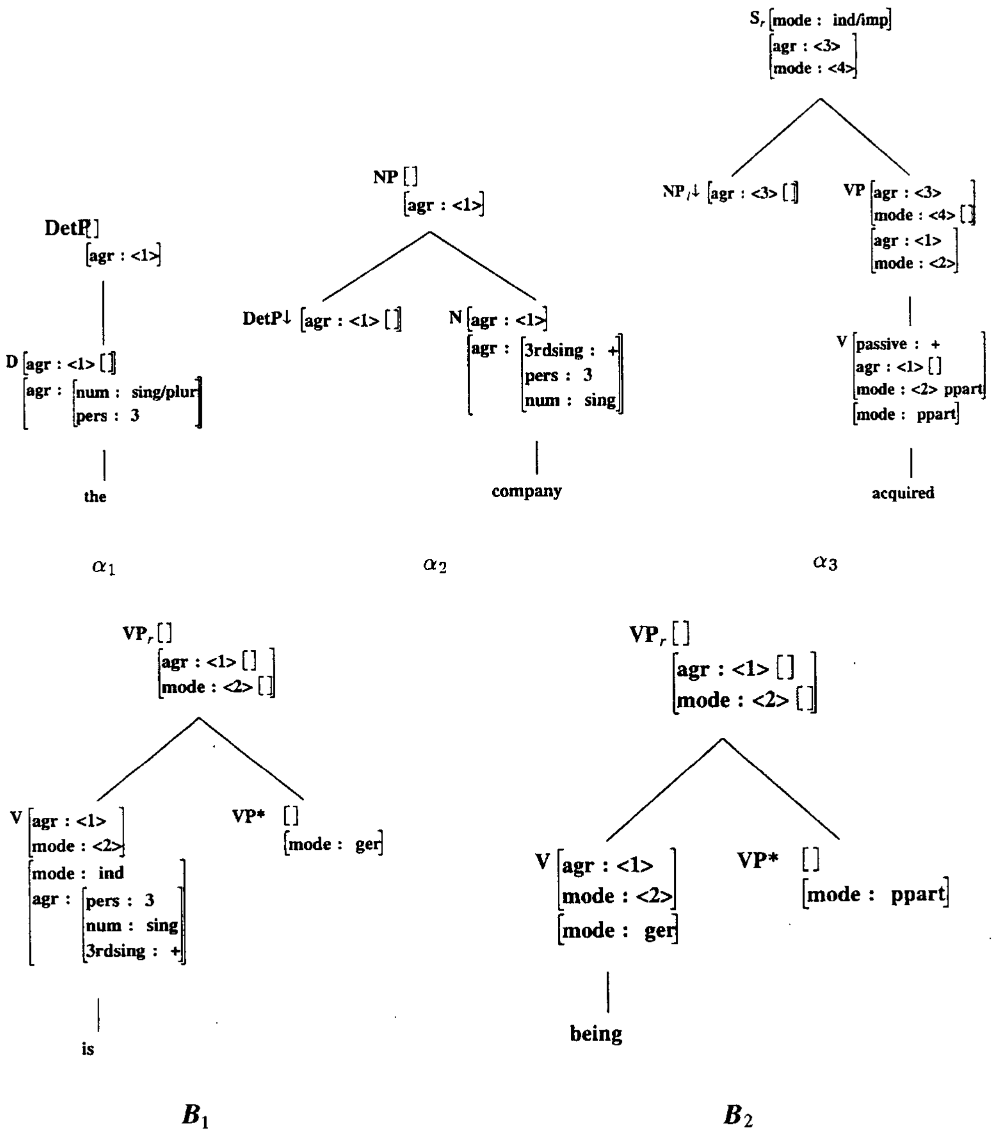

Figure 6
Elementary trees for the sentence: the company is being acquired.

the top and the bottom. The bottom FS contains information relating to the subtree rooted at the node, and the top FS contains information relating to the supertree at that node.[13] Features may get their values from three different sources:

- Morphology of anchor: from the morphological information of the lexical items that anchor the tree.
- Structural characteristics: from the structure of the tree itself (for

13 Nodes marked for substitution are associated with only the top FS.

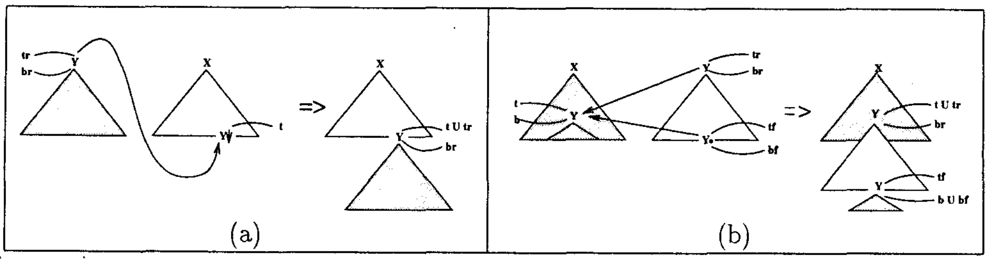

Figure 7
Substitution and adjunction in LTAG.

example, the *mode = ind/imp* feature on the root node in the α_3 tree in Figure 6).

- The derivation process: from unification with features from trees that adjoin or substitute.

Elementary trees are combined by substitution and adjunction operations. Substitution inserts elementary trees at the substitution nodes of other elementary trees. Figure 7(a) shows two elementary trees and the tree resulting from the substitution of one tree into the other. In this operation, a node marked for substitution in an elementary tree is replaced by another elementary tree whose root label matches the label of the node. The top FS of the resulting node is the result of unification of the top features of the two original nodes, while the bottom FS of the resulting node is simply the bottom features of the root node of the substituting tree.

In an adjunction operation, an auxiliary tree is inserted into an elementary tree. Figure 7(b) shows an auxiliary tree adjoining into an elementary tree and the result of the adjunction. The root and foot nodes of the auxiliary tree must match the node label at which the auxiliary tree adjoins. The node being adjoined to splits, and its top FS unifies with the top FS of the root node of the auxiliary tree, while its bottom FS unifies with the bottom FS of the foot node of the auxiliary tree. Figure 7(b) shows an auxiliary tree and an elementary tree, and the tree resulting from an adjunction operation. For a parse to be well-formed, the top and bottom FS at each node should be unified at the end of a parse.

The result of combining the elementary trees shown in Figure 6 is the derived tree, shown in Figure 8(a). The process of combining the elementary trees to yield a parse of the sentence is represented by the derivation tree, shown in Figure 8(b). The nodes of the derivation tree are the tree names that are anchored by the appropriate lexical items. The combining operation is indicated by the type of the arcs (a broken line indicates substitution and a bold line indicates adjunction) while the address of the operation is indicated as part of the node label. The derivation tree can also be interpreted as a dependency tree with unlabeled arcs between words of the sentence, as shown in Figure 8(c).

A broad-coverage grammar system, XTAG, has been implemented in the LTAG formalism. In this section, we briefly discuss some aspects related to XTAG for the sake of completeness. A more detailed report on XTAG can be found in XTAG-Group (1995). The XTAG system consists of a morphological analyzer, a part-of-speech tagger, a wide-coverage LTAG English grammar, a predictive left-to-right Earley-style parser for LTAG (Schabes 1990), and an X-windows interface for grammar development (Doran et al. 1994). The input sentence is subjected to morphological analysis

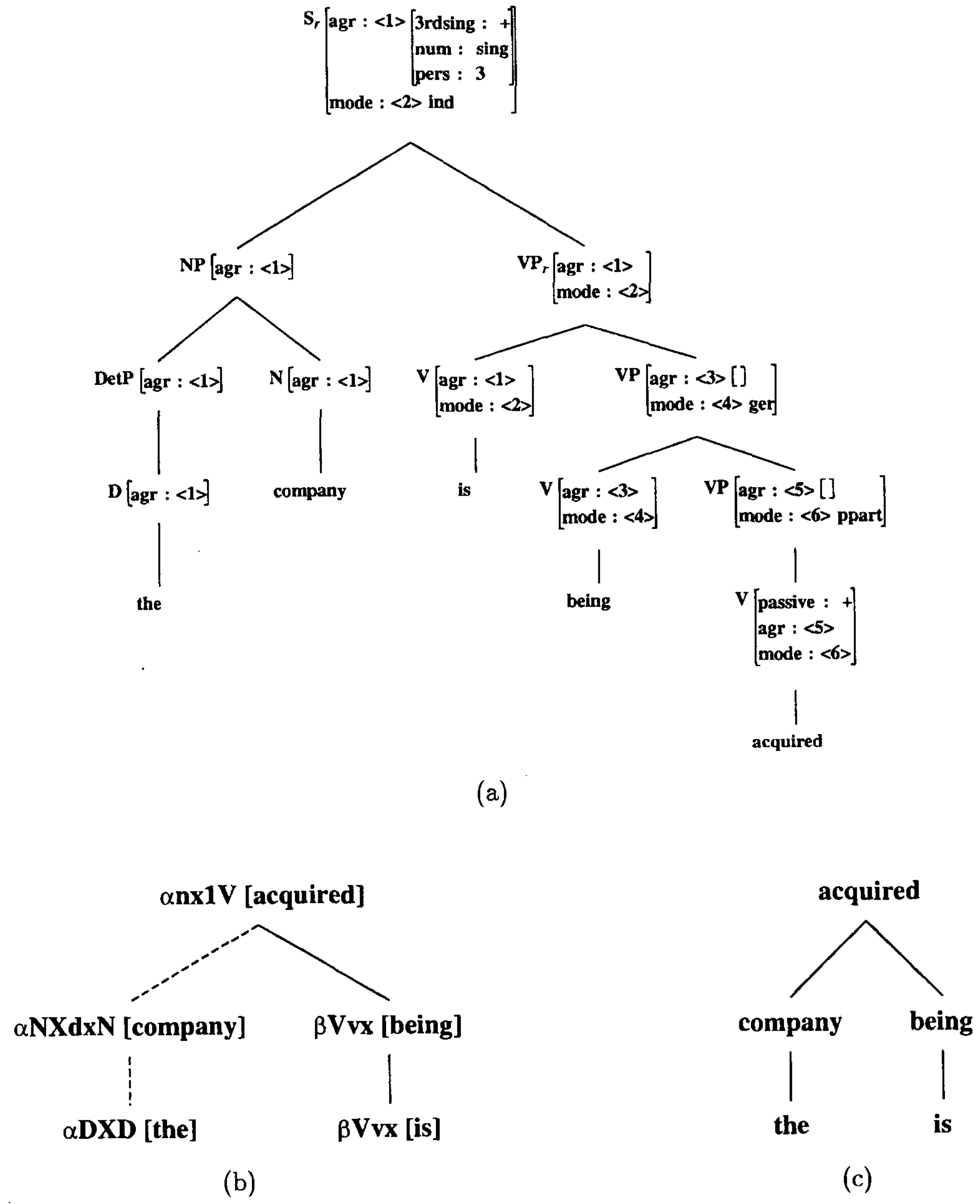
S_r [agr : <1> [3rdsing : + / num : sing / pers : 3] / mode : <2> ind]
NP [agr : <1>]
VP_r [agr : <1> / mode : <2>]
DetP [agr : <1>]
N [agr : <1>]
V [agr : <1> / mode : <2>]
VP [agr : <3> [] / mode : <4> ger]
D [agr : <1>]
company
is
V [agr : <3> / mode : <4>]
VP [agr : <5> [] / mode : <6> ppart]
the
being
V [passive : + / agr : <5> / mode : <6>]
acquired
(a)
αnx1V [acquired]
αNXdxN [company]
βVvx [being]
αDXD [the]
βVvx [is]
(b)
acquired
company
being
the
is
(c)

Figure 8
(a) Derived tree, (b) derivation tree, and (c) dependency tree for the sentence: the company is being acquired.

and is tagged with parts of speech before being sent to the parser. The parser retrieves the elementary trees that the words of the sentence anchor and combines them by adjunction and substitution operations to derive a parse of the sentence. The grammar of XTAG has been used to parse sentences from ATIS, IBM Manual and WSJ corpora (TAG-Group 1995). The resulting XTAG corpus contains sentences from these domains along with all the derivations for each sentence. The derivations provide predicate argument relationships for the parsed sentences.

Appendix B: Key Properties of LTAGs

In this section, we define the key properties of LTAGs: lexicalization, Extended Domain of Locality (EDL), and factoring of recursion from the domain of dependency (FRD), and discuss how these properties are realized in natural language grammars written in LTAGs. A more detailed discussion about these properties is presented in Joshi (1985, 1987), Kroch and Joshi (1985), Schabes, Abeillé, and Joshi (1988), and Joshi and Schabes (1996).

Definition
A grammar is lexicalized if it consists of:

- a finite set of elementary structures (strings, trees, directed acyclic graphs, etc.), each structure anchored on a lexical item.
- lexical items, each associated with at least one of the elementary structures of the grammar
- a finite set of operations combining these structures.

This property proves to be linguistically crucial since it establishes a direct link between the lexicon and the syntactic structures defined in the grammar. In fact, in lexicalized grammars all we have is the lexicon, which projects the elementary structures of each lexical item; there is no independent grammar.

Definition
The Extended Domain of Locality (EDL) property has two parts:

1. Every elementary structure must contain all and only the arguments of the anchor in the same structure.
2. For each lexical item, the grammar must contain an elementary structure for each syntactic environment the lexical item might appear in.

Part (1) of EDL allows the anchor to impose syntactic and semantic constraints on its arguments directly since they appear in the same elementary structure that it anchors. Hence, all elements that appear within one elementary structure are considered to be local. This property also defines how large an elementary structure in a grammar can be. Figure 9 shows trees for the following example sentences:

(1) John seems to like Mary.

(2) John hit Mary.

(3) Who did John hit?

Figure 9(a) shows the elementary tree anchored by *seem* that is used to derive a raising analysis for sentence 1. Notice that the elements appearing in the tree are only those that serve as arguments to the anchor and nothing else. In particular, the subject NP

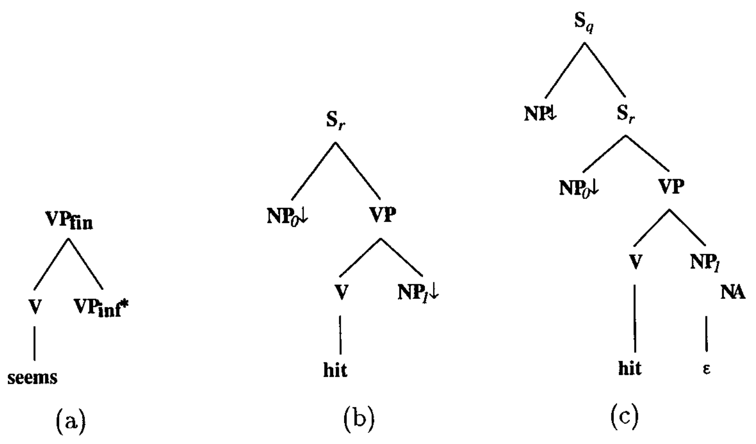

Figure 9
(a) Tree for raising analysis, anchored by seems; (b) transitive tree; (c) object extraction tree for the verb hit.

(*John* in sentence 1) does not appear in the elementary tree for *seem* since it does not serve as an argument for *seem*. Figure 9(b) shows the elementary tree anchored by the transitive verb *hit* in which both the subject NP and object NP are realized within the same elementary tree.

LTAG is distinguished from other grammar formalisms by possessing part (2) of the EDL property. In LTAGs, there is one elementary tree for every syntactic environment that the anchor may appear in. Each elementary tree encodes the linear order of the arguments of the anchor in a particular syntactic environment. For example, a transitive verb such as *hit* is associated with both the elementary tree shown in Figure 9(b) for a declarative transitive sentence such as sentence 2, and the elementary tree shown in Figure 9(c) for an object extracted transitive sentence such as sentence 3. Notice that the object noun phrase is realized to the left of the subject noun phrase in the object extraction tree.

As a consequence of the fact that LTAGs possess the part (2) of the EDL property, the derivation structures in LTAGs contain the information of a dependency structure. Another aspect of EDL is that the arguments of the anchor can be filled in any order. This is possible because the elementary structures allocate a slot for each argument of the anchor in each syntactic environment that the anchor appears in.

There can be many ways of constructing the elementary structures of a grammar so as to possess the EDL property. However, by requiring that the constructed elementary structures be "minimal," the third property of LTAGs namely, factoring of recursion from the domain of dependencies, follows as a corollary of EDL.

Definition
Factoring of recursion from the domain of dependencies (FRD): Recursion is factored away from the domain for the statement of dependencies.

In LTAGs, recursive constructs are represented as auxiliary trees. They combine with elementary trees by the operation of adjunction. Elementary trees define the domain for stating dependencies such as agreement, subcategorization, and filler-gap dependencies. Auxiliary trees, by adjunction to elementary trees, account for the long-distance behavior of these dependencies.

An additional advantage of a grammar possessing FRD and EDL properties is that feature structures in these grammars are extremely simple. Since the recursion has been factored out of the domain of dependency, and since the domain is large enough for agreement, subcategorization, and filler-gap dependencies, feature structures in such systems do not involve any recursion. In fact they reduce to typed terms that can be combined by simple term-like unification.

Acknowledgments
This work was done when the first author was at the University of Pennsylvania. It was partially supported by NSF grant NSF-STC SBR 8920230, ARPA grant N00014-94 and ARO grant DAAH04-94-G0426. We would like to thank Steven Abney, Raman Chandrasekar, Christine Doran, Beth Ann Hockey, Mark Liberman, Mitch Marcus, and Mark Steedman for useful comments and discussions which have helped shape this work. We also thank the reviewers for their insightful comments and suggestions to improve an earlier version of this paper.

References

Abney, Steven. 1990. Rapid incremental parsing with repair. In *Proceedings of the 6th New OED Conference: Electronic Text Research*, pages 1–9, University of Waterloo, Waterloo, Ontario, Canada.

Alshawi, Hiyan and David Carter. 1994. Training and scaling preference functions for disambiguation. *Computational Linguistics*, 20(4):635–648.

Appelt, D., J. Hobbs, J. Bear, D. J. Israel, and M. Tyson. 1993. FASTUS: A finite-state processor for information extraction from real-world text. In *Proceedings of IJCAI-93*, Chambery, France, September.

Black, Ezra, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. 1993. Towards History-based Grammars: Using Richer Models for Probabilistic Parsing. In *Proceedings of the 31st Annual Meeting*, pages 31–37, Columbus, OH. Association for Computational Linguistics.

Bod, Rens and Ronald Kaplan. 1998. A probabilistic corpus-driven model for lexical-functional analysis. In *Proceedings of COLING-ACL '98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montreal, Quebec, Canada, August.

Brill, Eric. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Annual Meeting*, Columbus, OH. Association for Computational Linguistics.

Chandrasekar, R., Christine Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, Copenhagen, Denmark, August.

Chandrasekar, R. and B. Srinivas. 1997a. Automatic induction of rules for text simplification. *Knowledge-based Systems*, 10:183–190.

Chandrasekar, R. and B. Srinivas. 1997b. Gleaning information from the web: Using syntax to filter out irrelevant information. In *Proceedings of AAAI 1997 Spring Symposium on NLP on the World Wide Web.*

Chandrasekar, R. and B. Srinivas. 1997c. Using supertags in document filtering: The effect of increased context on information retrieval effectiveness. In *Proceedings of Recent Advances in NLP (RANLP) '97*, Tzigov Chark, Bulgaria, September.

Chandrasekar, R. and B. Srinivas. 1997d. Using syntactic information in document filtering: A comparative study of part-of-speech tagging and supertagging. In *Proceedings of RIAO'97*, Montreal, Quebec, Canada, June.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence AAAI*, pages 47–66, Menlo Park, CA.

Chomsky, Noam. 1992. *A Minimalist Approach to Linguistic Theory*. MIT Working Papers in Linguistics, Occasional Papers in Linguistics, No. 1.

Church, Kenneth Ward. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *2nd Applied Natural Language Processing Conference*, pages 136–143, Austin, TX.

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

Doran, Christine, Dania Egedi, Beth Ann

Hockey, B. Srinivas, and Martin Zaidel. 1994. XTAG System—A wide coverage grammar for English. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, August.

Doran, Christine, Michael Niv, Breckenridge Baldwin, Jeffrey Reynar, and B. Srinivas. 1997. Mother of Perl: A Multi-tier pattern description language. In *Proceedings of the International Workshop on Lexically Driven Information Extraction*, Frascati, Italy, July.

Doran, Christine and B. Srinivas. 1994. A wide-coverage CCG parser. In *Proceedings of the 3rd TAG+ Conference*, Paris, France.

Fujisaki, T., F. Jelinek, J. Cocke, E. Black, and T. Nishino. 1989. A probabilistic parsing method for sentence disambiguation. In *Proceedings of the 1st Annual International Workshop of Parsing Technologies*, Pittsburgh, PA.

Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, MA.

Good, I. J. 1953. The population frequenceis of species and the estimation of population parameters. *Biometrika 40 (3 and 4)*, pages 237–264.

Grishman, Ralph. 1995. Where's the syntax? The New York University MUC-6 System. In *Proceedings of the Sixth Message Understanding Conference*, Columbia, MD.

Gross, Maurice. 1984. Lexicon-grammar and the syntactic analysis of French. In *Proceedings of the 10th International Conference on Computational Linguistics (COLING'84)*, Stanford, CA.

Hobbs, Jerry R., Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1997. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In E. Roche and Y. Schabes, editors, *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, MA.

Hobbs, Jerry R., Douglas E. Appelt, John Bear, David Israel, Andy Kehler, Megumi Kamayama, David Martin, Karen Myers, and Mabry Tyson. 1995. SRI International FASTUS system MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference*, Columbia, MD.

Jelinek, Fred, John Lafferty, David M. Magerman, Robert Mercer, Adwait Ratnaparkhi, and Salim Roukos. 1994. Decision tree parsing using a hidden derivation model. In *Proceedings from the ARPA Workshop on Human Language Technology Workshop*, March.

Joshi, Aravind K. 1960. Computation of syntactic structure. In *Advances in Documentation and Library Science*, volume III, Part 2. Interscience Publishers, Inc., NY.

Joshi, Aravind K. 1985. Tree adjoining grammars: How much context sensitivity is required to provide a reasonable structural description? In D. Dowty, I. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*. Cambridge University Press, Cambridge, U.K., pages 206–250.

Joshi, Aravind K. 1987. An introduction to tree adjoining grammars. In A. Manaster Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

Joshi, Aravind K. 1998. Role of constrained computational systems in natural language processing. *Artificial Intelligence*, 103:117–132.

Joshi, Aravind K. and Philip Hopely. 1997. A parser from antiquity. *Natural Language Engineering*, 2(4).

Joshi, Aravind K., L. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*.

Joshi, Aravind K. and Yves Schabes, 1996. Tree-adjoining grammars. In *Handbook of Formal Languages and Automata*. Springer-Verlag, Berlin.

Joshi, Aravind K. and B. Srinivas. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, August.

Kaplan, Ronald and Joan Bresnan. 1983. Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

Karlsson, F., A. Voutilainen, J. Heikkilä, and A. Anttila. 1994. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin and NY.

Karttunen, L. J-P. Chanod, G. Grefenstette, and A. Schiller. 1997. Regular expressions for language engineering. *Natural Language Engineering*, 2(4).

Kasper, Robert, Bernd Kiefer, Klaus Netter, and K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. In *Proceedings of the 33rd Annual Meeting*, Cambridge, MA. Association for Computational Linguistics.

Katz, Slava M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics,* *Speech and SignalProcessing*, 35(3):400–401.

Kempe, Andre. 1994. Probabilistic Tagging with Feature Structures. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, August.

Kroch, Anthony S. and Aravind K. Joshi. 1985. The linguistic relevance of tree adjoining grammars. Technical Report MS-CIS-85-16, Department of Computer and Information Science, University of Pennsylvania.

Kuno, S. 1966. Harvard predictive analyzer. In David G. Hays, editor, *Readings in Automatic Language Processing*. American Elsevier Pub. Co., NY.

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

Marcus, Mitchell M., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Nagao, Makoto. 1994. Varieties of heuristics in sentence processing. In *Current Issues in Natural Language Processing: In Honour of Don Walker*. Giardini with Kluwer.

Ney, Herman, Ute Essen, and Reinhard Kneser. 1995. On the estimation of 'small' probabilities by leaving-one-out. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2).

Niesler, T. R. and P. C. Woodland. 1996. A variable-length category-based n-gram language model. In *Proceedings, IEEE ICASSP*.

Pollard, Carl and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics. Vol. 1: Fundamentals*. CSLI.

Roche, Emmanuel. 1993. *Analyse syntaxique transformationelle du francais par transducteurs et lexique-grammaire*. Ph.D. thesis, Universite Paris 7.

Schabes, Yves. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, Computer Science Department, University of Pennsylvania.

Schabes, Yves, Anne Abeillé, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August.

Schabes, Yves and Aravind K. Joshi. 1991. Parsing with lexicalized tree adjoining grammar. In M. Tomita, editor, *Current Issues in Parsing Technologies*. Kluwer Academic Publishers.

Schabes, Y., M. Roth, and R. Osborne. 1993. Parsing the Wall Street Journal with the inside-outside algorithm. In *Proceedings of the European ACL*.

Sleator, Daniel and Davy Temperley. 1991. Parsing English with a Link Grammar. Technical Report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University.

Srinivas, B. 1996. "Almost parsing" technique for language modeling. In *Proceedings of ICSLP96 Conference*, Philadelphia, PA.

Srinivas, B. 1997a. *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. Ph.D. thesis, University of Pennsylvania.

Srinivas, B. 1997b. Performance evaluation of supertagging for partial parsing. In *Proceedings of the International Workshop on Parsing Technologies*, September.

Srinivas, B., Christine Doran, and Seth Kulick. 1995. Heuristics and parse ranking. In *Proceedings of the 4th Annual International Workshop on Parsing Technologies*, Prague, September.

Steedman, Mark. 1987. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5:403–439.

Steedman, Mark, editor. 1997. *The Syntactic Interface*. MIT Press, Cambridge, MA and London, England.

Tapanainen, Pasi and Timo Järvinen. 1994. Syntactic analysis of natural language using linguistic rules and corpus-based patterns. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, August.

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

Vijay-Shanker, K. and Aravind K. Joshi. 1991. Unification based tree adjoining grammars. In J. Wedekind, editor, *Unification-based Grammars*. MIT Press, Cambridge, MA.

Voutilainen, Atro. 1994. *Designing a Parsing Grammar*. Publications of the Department of General Linguistics, University of Helsinki.

Waltz, D. 1975. Understanding line drawings of scenes with shadows. In P. Winston, editor, *Psychology of Computer Vision*, MIT Press.

Weischedel, Ralph, Richard Schwartz, Jeff Palmucci, Marie Meteer, and Lance Ramshaw. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):359–382, June.

XTAG-Group, The. 1995. A lexicalized tree adjoining grammar for English. Technical Report IRCS 95-03, University of Pennsylvania.