Performance Evaluation of Supertagging for Partial Parsing

B. Srinivas
Dept. of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
srini@linc.cis.upenn.edu

Abstract

In previous work we introduced the idea of supertagging as a means of improving the efficiency of a lexicalized grammar parser. In this paper, we present supertagging in conjunction with a lightweight dependency analyzer as a robust and efficient partial parser. The present work is significant for two reasons. First, we have vastly improved our results; 92% accurate for supertag disambiguation using lexical information, larger training corpus and smoothing techniques. Second, we show how supertagging can be used for partial parsing and provide detailed evaluation results for detecting noun chunks, verb chunks, preposition phrase attachment and a variety of other linguistic constructions. Using supertag representation, we achieve a recall rate of 93.0% and a precision rate of 91.8% for noun chunking, improving on the best known result for noun chunking.

1 Introduction

A number of grammar formalisms such as HPSG [Pollard and Sag, 1987], CCG [Steedman, 1987], Lexicon-Grammars [Gross, 1984], LTAG [Schabes et al., 1988], Link Grammars [Sleator and Temperley, 1991] fall into the class of lexicalized grammar formalisms. Lexicalized grammar formalisms associate increasingly rich and complex descriptions with each lexical item. Typically, a lexical item in these frameworks is associated with more than one description. The task of a parser for such formalisms can be viewed as first selecting the appropriate description for individual words given the context of the input and then combining them to arrive at a description for the entire input.

In [Joshi and Srinivas, 1994], we introduced the idea of supertagging as a means of selecting the appropriate descriptions for each word given the context of a sentence, even before parsing begins, so as to improve the efficiency of a lexicalized grammar parser. In this paper, we present supertagging in conjunction with a lightweight dependency analyzer as a robust and efficient partial parser. We provide detailed evaluation results of using supertag representation for detecting noun chunks, verb chunks, preposition phrase attachment, appositives and parenthetical constructions. Using supertags, we achieve a recall rate of 93.0% and a precision rate of 91.8% for noun chunking, improving on the best known result for noun chunking [Ramshaw and Marcus, 1995]. We also present vastly improved supertag disambiguation results from previously published 68% accurate to 92% accurate, a significant result keeping in mind that the supertags contain richer information than part-of-speech tags.

The outline of this paper is as follows. In Section 2, we present a brief introduction to Lexicalized Tree-Adjoining Grammars. In Section 3, we review the notion of supertags and in Section 4, we discuss the details of the trigram model for disambiguating supertags and the results of evaluation on the Wall Street Journal corpus. In Section 5, we introduce the lightweight dependency analyzer. A detailed evaluation of the supertag and lightweight dependency analyzer system is presented in Section 6. In Section 7, we briefly discuss two new models for supertagging.

2 Lexicalized Tree-Adjoining Grammars

Each elementary tree of Lexicalized Tree-Adjoining Grammar (LTAG)[Joshi, 1985, Schabes et al., 1988] is associated with at least one lexical item called the *anchor* of that tree. All the arguments of the anchor are realized

as substitution or adjunction slots within an elementary tree. Thus an elementary tree serves as a complex description of the anchor and provides a domain of locality over which the anchor specifies syntactic and semantic (predicate-argument) constraints. Elementary trees are of two types: *initial trees* (α trees in Figure 2) that represent non-recursive linguistic structures such as NPs, PPs and *auxiliary trees* (β trees in Figure 2) that represent recursive structures which are adjuncts to basic structure (e.g. relative clauses, sentential adjuncts, adverbials).

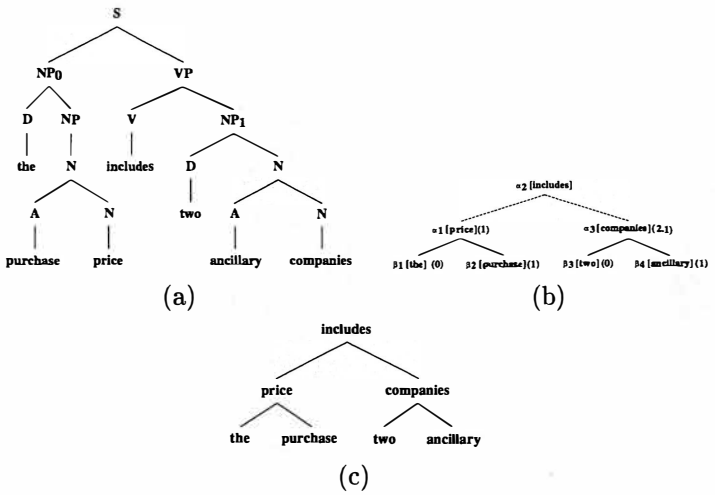

Figure 1: (a): Derived tree (b): Derivation structure (c): Dependency tree for the sentence *the purchase price includes two ancillary companies*

Elementary trees are combined by two operations, *substitution* and *adjunction*. The result of combining the last row of elementary trees of Figure 2 is the *derived tree* (Figure 1(a)). But the more important structure in an LTAG parse is the *derivation tree* (Figure 1(b)) which represents the process of combining the elementary trees to yield a parse. The derivation tree can also be interpreted as a *dependency tree* (Figure 1(c)) with unlabeled arcs between words of the sentence. A wide-coverage English grammar called XTAG has been implemented in the LTAG framework. This grammar has been used to parse sentences from the Wall Street Journal, IBM manual and ATIS domains. A detailed description of this system and its performance results are presented in [Doran et al., 1994].

3 Supertags

The elementary trees of LTAG localize dependencies, including long distance dependencies, by requiring that all and only the dependent elements be present within the same tree. As a result of this localization, a lexical item may be (and almost always is) associated with more than one elementary tree. The example in Figure 2 illustrates the set of elementary trees assigned to each word of the sentence *the purchase price includes two ancillary companies*. We call these elementary trees *supertags*, since they contain more information (such as subcategorization and agreement information) than standard part-of-speech tags. Supertags for recursive and non-recursive constructs are labeled with βs and αs respectively.

The task of a lexicalized grammar parser can be viewed as a two step process. The first step is to select the appropriate supertags for each word of the input and the second step is to combine the selected supertags with substitution and adjunction operations. We call the first step as *Supertagging*. Note that, as in standard part-of-speech disambiguation, supertagging could have been done by a parser. However, just as carrying out part-of-speech disambiguation prior to parsing makes the job of the parser much easier and therefore run faster, supertagging reduces the work of the parser even further.

More interesting is the fact that the result of supertagging is almost a parse in the sense that the parser need 'only' link the individual structures to arrive at a complete parse. We present such a simple linking procedure (*Lightweight Dependency Analyzer*) in Section 5. This method can also be used to parse sentence fragments where it is not possible to combine the disambiguated supertag sequence into a single structure.

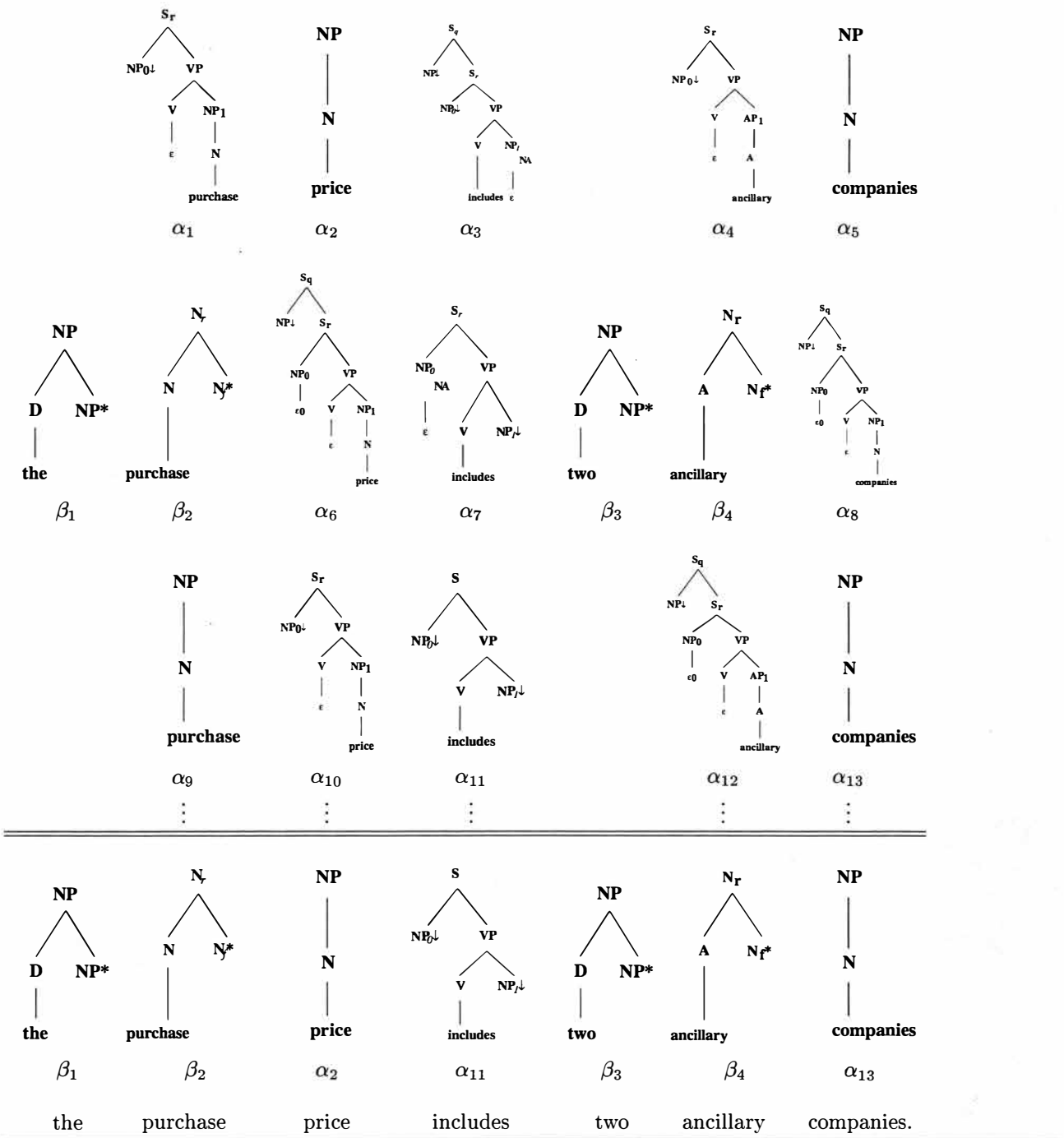

Figure 2: A selection of the supertags associated with each word of the sentence *the purchase price includes two ancillary companies*

4 Trigram Model for Supertagging

The task of supertagging is similar to part-of-speech tagging in that, given a set of tags for each word, the objective is to assign the appropriate tag to each word based on the context of the sentence. Owing to this similarity of supertagging to part-of-speech tagging, we use a trigram model [Church, 1988, Weischedel et al., 1993] to disambiguate supertags. The objective in a trigram model is to assign the most probable supertag sequence for a sentence given the approximation that the supertag for the current word is only influenced by the lexical preference of the current word and the contextual preference based on the supertags of the preceding two words.

Although it is quite evident, owing to the rich information present in supertags, that the dependencies between supertags can easily span beyond the trigram context, one of the goals of this work is to explore the limits of the trigram tagging approach. It appears that a CKY style dynamic programming model that takes advantage of the dependency requirements of each supertag may perform better for supertag disambiguation. However, such an approach is too much like parsing and the objective here is to see how much disambiguation can be done without really parsing.

The lexical and contextual preferences for the trigram model are estimated from a corpus of sentences where the words are tagged with the correct supertag. The estimates for unseen events are arrived at using a smoothing technique. We use Good-Turing discounting technique [Good, 1953] combined with Katz's back-off model for smoothing. 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 unknown word probability. In conjunction with the word features, we exploit the organization of the supertags. The supertags are organized so that transformationally related supertags (indicative, passives, relative clauses, extraction supertags) are grouped into a single "family". Using this notion, if a word w_i in the training material appears with a supertag t_i which belongs to a tree family T, then w_i is associated with all the other members of the tree family T.

Experiments and Results

As reported in the COLING'94 paper [Joshi and Srinivas, 1994], we experimented with a trigram model for supertagging that was trained on (part-of-speech, supertag) pairs collected from the LTAG derivations of 5000 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 making the model lexically sensitive, using a larger training set and incorporating smoothing techniques.

Table 1 shows the performance of the trigram model that was trained on two sets of Wall Street Journal data, 200K words[1] and 1000K words[2] and tested on 50K words[3]. The Treebank parses for the training and test sentences were converted into supertag representation using heuristics specified over parse tree contexts (parent, grandparent, children and sibling information)[4]. A total of 300 different supertags were used in these experiments. Supertag performance is measured as the percentage of words that are correctly supertagged by the model when compared against the supertags for the words in the test corpus.

Size of training set	Training	Size of test set	% Correct
200K	Unigram (Baseline)	50K	75.3%
	Trigram	50K	90.9%
1000K	Unigram (Baseline)	50K	77.2%
	Trigram	50K	92.2%

Table 1: Performance of the supertagger on the WSJ corpus

As mentioned earlier, 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. Alternatively, the dependency information

[1] Sentences in wsj_15 through wsj_18 of Penn Treebank.
[2] Sentences in wsj_00 through wsj_24, except wsj_20 of Penn Treebank.
[3] Sentences in wsj_20 of Penn Treebank.
[4] An example of the heuristics is given in [Srinivas, 1997]

encoded in the supertagger can be used in conjunction with a simple linking procedure as a robust, fast and efficient partial parser. Such an approach can also be used to parse sentence fragments where it is not possible to combine the disambiguated supertag sequence into a single structure.

5 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 within the supertag. Using the structural information coupled with the argument requirements of a supertag, a simple algorithm such as the one below provides a method for annotating the sentence with dependency links.

Step 1: `For each modifier supertag` s `in the sentence`
`Compute the dependencies for` s
`Mark the words serving as complements as unavailable for step 2.`

Step 2: `For the non-recursive supertags` s `in the sentence`
`Compute the dependencies for` s

Compute Dependencies for s_i of w_i:
`For each slot` d_{ij} `in` s_i `do`
`Connect word` w_i `to the nearest word` w_k `to the left or right of` w_i `depending on the direction of` d_{ij}`, skipping over marked supertags if any, such that` $d_{ij} \in$ `internal_nodes(`s_k`)`

An example illustrating the output from this algorithm is shown in Table 2. The first column lists the word positions in the input, the second column lists the words, the third lists the names of the supertags assigned to each word by a supertagger. The slot requirement of each supertag is shown in column four and the dependency links among the words, computed by the above algorithm, is shown in the fifth column. The $*$ and the . beside a number indicate the type of the dependency relation, $*$ for modifier relation and . for complement relation.

Position	Word	Supertag	Slot req.	Dependency links
0	The	β_1	+NP$*$	2$*$
1	purchase	β_2	+N$*$	2$*$
2	price	α_2	–	
3	includes	α_{11}	–NP. +NP.	2. 6.
4	two	β_3	+NP$*$	6$*$
5	ancillary	β_4	+N$*$	6$*$
6	companies	α_{13}	–	

Table 2: An example sentence with the supertags assigned to each word and dependency links among words

6 Evaluation of Supertag and LDA system

Due to the fact that our system produces a dependency annotated sentence as the output of the parsing process, parsing evaluation metrics that measure the performance of constituent bracketing such as Parseval [Harrison et al., 1991] are unsuitable. Although the supertags contain constituent information, and it is possible to convert a dependency linkage into a constituency based parse, the parseval metric does not have provision for evaluating unrooted parse trees since the output of our system can result in disconnected dependency linkages.

In contrast, we evaluate the performance of our system in terms of its ability to identify certain linguistic structures such as noun groups, verb groups, preposition attachments, appositive and parenthetical constructions. We compare the performance of our system to the performance of other systems specifically designed to identify these structures, when available. The results presented in this section are based on a trigram supertagger trained on 200K words of wsj_15 through wsj_18 and tested on 50K words from wsj_20.

6.1 Text Chunking using Supertags

We have applied the supertag and Lightweight Dependency Analyzer (LDA) system for text chunking. Text chunking, proposed by Abney [Abney, 1991] involves partitioning a sentence into a set of non-overlapping segments. Text chunking serves as a useful and relatively tractable precursor to full parsing. Text chunks primarily consist of non-recursive noun and verb groups. The chunks are assumed to be minimal and hence non-recursive in structure.

Noun Chunking: Supertag based text chunking is performed by the local application of functor-argument information encoded in supertags. Once the head noun of the noun chunk is identified, the prenominal modifiers that are functors of the head noun or functors of the functors of the noun are included in the noun chunk. The head of a noun phrase is identified as the noun with a particular initial (non recursive) supertag (α) in the grammar. Based on that simple algorithm, we can identify noun chunks, for example, shown in (1) and (2).

(1) its increasingly rebellious citizens

(2) two $ 400 million real estate mortgage investment conduits

Ramshaw and Marcus [Ramshaw and Marcus, 1995] present a transformation-based noun chunker that uses a learning scheme presented in [Brill, 1993]. The performance of this noun chunker using rules defined with and without rules incorporating lexical information is shown in Table 3.

System	Training Size	Recall	Precision
R&M	Baseline	81.9%	78.2%
R&M (without lexical information)	200K	90.7%	90.5%
R&M (with lexical information)	200K	92.3%	91.8%
Supertags	Baseline	74.0%	58.4%
Supertags	200K	93.0%	91.8%

Table 3: Performance comparison of the transformation based noun chunker and the supertag based noun chunker

Table 3 also shows the performance of the supertag based noun chunking, trained and tested on the same texts as the transformation-based noun chunker. The supertag-based noun chunker performed better than transformation-based noun chunker with lexical templates Moreover, the supertag-based noun chunking not only identifies the extents of the noun chunks but by the virtue of the functor-argument information, provides internal structure to the noun chunks. This internal structure of the noun chunks could be utilized during subsequent linguistic processing.

Verb Chunking: We also performed an experiment similar to noun chunking to identify verb chunks. We treat a sequence of verbs and verbal modifiers, including auxiliaries, adverbs, modals as constituting a verb group (shown in (3) and (4)). Similar to noun chunking, verb chunking can be performed using local functor-argument information encoded in supertags. However, for verb chunks, the scan is made from left to right starting with a verbal modifier supertag, either an auxiliary verb or an adverbial supertag and including all functors of a verb or a verb modifier. The verb chunker performed at 86.5% recall and 91.4% precision.

(3) would not have been stymied

(4) just beginning to collect

Preposition Phrase Attachment

Supertags distinguish a noun-attached preposition from a verb-attached preposition in a sentence, as illustrated by the two different supertags in (5) and (6). Due to this distinction, the supertagging algorithm can be evaluated based on its ability to select the correct supertag for the prepositions in a text.

(5) sell 500 railcar platforms to/B_vxPnx Trailer Train Co. of/B_nxPnx Chicago

(6) begin delivery of/B_nxPnx goods in/B_vxPnx the first quarter

The task of preposition attachment has been worked on by a number of researchers in the past. Humans perform only at an accuracy of 88% accuracy [Ratnaparkhi et al., 1994] which gives an indication of the complexity of the task. Table 4 presents a comparative evaluation of various approaches in the literature against the supertag based approach, for the preposition attachment task.

System	Accuracy
Ratnaparkhi, Reynar & Roukos	81.6%
Hindle & Rooth	78-80%
Brill & Resnik	81.9%
Collins & Brooks	84.5%
Supertags	81.1%

Table 4: Performance comparison of the supertag based preposition phrase attachment against other approaches to preposition phrase attachment

It must be pointed out that the supertagger makes the preposition attachment decision (choosing between B_vxPnx and B_nxPnx) based on a trigram context. Most often than not the preposition is not within the same trigram window as the noun and the verb due to modifiers of the noun and the verb. However, despite this limitation, the trigram approach performs as well as Ratnaparkhi, Reynar & Roukos [Ratnaparkhi et al., 1994] and Hindle and Rooth [Hindle and Rooth, 1991], and is outperformed only by methods (Brill and Resnik [Brill and Resnik, 1994], Collins and Brooks[Collins and Brook, 1995]) that use four words: the verb, the object noun, the preposition and the complement of the preposition in making the attachment decision.

Other Constructions

In this section, we summarize the performance of the supertagger in identifying constructions such as appositives, parentheticals, relative clauses and coordinating conjunctions.

Appositives: In the XTAG grammar, the appositive construction has been analyzed using a Noun Phrase modifying supertag that is anchored by a comma which takes the appositive Noun Phrase as an argument; for example, the comma in (7) and the second comma in (8) anchor appositive supertags. Thus, a comma in a sentence could either anchor an appositive supertag or a set of coordination supertags, one supertag for each type of coordination. Further, a comma also anchors supertags that mark parentheticals as in (9). The task of the supertagger is to disambiguate among the various supertags and assign the appropriate supertag to the comma, in the appositive context. In Table 5, we present the performance results of the supertagger on such a task. The baseline of assigning the most likely supertag to comma results in zero percent recall.

Parentheticals: Propositional attitude verbs such as *say* and *believe* can not only appear in the canonical word order of subject-verb-complement, but can also appear at other positions in a sentence, as in (8) and (9). Also, the relative order of the subject and the verb can also be reversed as in (8). The XTAG grammar distinguishes such adverbial constructions from the sentence complement construction by assigning different supertags to the verb. A detailed analysis of this construction is presented in [Doran, 1996]. The task of the supertagger is to disambiguate among the sentence complement supertag and the various adverbial supertags for the verb given the context of the sentence. Table 5 presents the performance results of the supertagger on this task. Once again the baseline of selecting the most likely supertag of the verb results in zero percent recall.

(7) Van Pell , 44

(8) " The U.S. underestimated Noriega all along , " *says* Ambler Moss , a former Ambassador to Panama

(9) Mr. Noriega 's relationship to American intelligence agencies became contractual in either 1966 or 1967 , intelligence officials *say* .

Construction	# of occurrences	# identified	# correct	Recall	Precision
Appositive	362	491	306	84.5%	62.3%
Parentheticals	225	200	170	75.5%	85%
Coordination	1886	1750	1329	70.5%	75.9%
Relative Clauses	297	269	116	39.0%	43.2%
Relative Clauses (ignoring valency)	297	269	156	52.5%	58%

Table 5: Performance of the trigram supertagger on Appositive, Parentheticals, Coordination and Relative Clause constructions.

Coordination Conjunctions: In Table 5, we also present the performance of the supertagger in identifying coordination constructions. Coordination conjunctions anchor a supertag that requires two conjuncts, one on either side of the anchor. There is one supertag for every possible pair of conjunct types. We not only have supertags that coordinate like types but we also include supertags that coordinate unlike types amounting to about 30 different supertags for coordination.

Relative Clauses: Due to extended domain of locality of LTAGs, verbs are associated with one relative clause supertag for each of the arguments of the verb. The task of the supertagger in LTAG is not only to identify that the verb is in a relative clause structure but also to identify the valency of the verb and the argument that is being relativized. The performance of the supertagger with and without the valency information being taken into account is present in Table 5.

We are unaware of any other quantitative results on WSJ data for identifying these constructions, for comparative evaluation. It is interesting to note that the performance of the supertagger in identifying appositives and parenthetical construction is better than for coordination conjunctions and relative clause constructions. We believe that this might in part be due to the fact that Appositives and Parentheticals can mostly be disambiguated using relatively local contexts. In contrast, disambiguation of Coordinate constructions and Relative clauses typically require large contexts that are not available for a trigram supertagger.

6.2 Performance of Supertag and LDA system

In this section, we present results from two experiments using the supertagger in conjunction with the LDA system as a dependency parser. In order to evaluate the performance of this system, we need a dependency annotated corpus. However, the only annotated corpora we are aware of, the Penn Treebank [Marcus et al., 1993] and the SUSANNE corpus [Sampson, 1994], annotate sentences with constituency trees. In the interest of time and the need for a dependency annotated corpus, we decided to transform the constituency trees of the two corpora using some rules and heuristics. It must be noted that the resulting corpora is only but an approximation to a manually annotated dependency corpus. However, although the annotation resulting from the transformation does not conform to a standard dependency annotation, it is nevertheless an invaluable resource for performance evaluation of dependency parsers.

For each constituent of a parse a head word is associated using the head percolation table introduced in [Jelinek et al., 1994, Magerman, 1995]. The head percolation table associates with each possible constituent label an ordered list of possible children of the constituent whose head word is passed to the parent. The annotation of a parse tree with head word information proceeds bottom up. At any constituent, the percolation information is consulted to determine which of the constituent's children would pass the head word over to their parent. Once the head words are annotated, the dependency notation is generated by making the head words of non-head constituents to be dependent on the head of the head constituent. In a similar manner to the WSJ conversion process, we converted the subset of the LOB annotation of the SUSANNE corpus into a dependency notation.

In the first experiment, we use the dependency versions of the Penn Treebank annotation of the WSJ corpus and the LOB annotation of the SUSANNE corpus as gold standards. However, in the second experiment, we

use derivation structures of WSJ sentences that were parsed using the XTAG system as the gold standard. The derivation structures serve as dependency structures that are closest in conventions to those assumed by the LDA system.

Experiment 1:

The trigram supertagger trained on 200,000 words of the WSJ corpus was used in conjunction with the LDA to provide a dependency analysis for 2000 sentences of Section 20 of the WSJ corpus. The Penn Treebank parses for these sentences were converted into dependency notation which was used as the gold standard. A dependency link produced by the LDA was regarded to be correct if the words being related by the link were also present in the gold standard. However, to account for the differences in the annotation, the dependency relations among words in a noun group were treated as equivalent to each other. So also, dependency relations among words in a verb group were treated as equivalent to each other. There were 47,333 dependency links in the gold standard and the LDA 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%.

We conducted a similar experiment using the SUSANNE corpus. Using the same trigram supertagger trained on the 200,000 words of the WSJ corpus in conjunction with the LDA, we provided a dependency analysis for the sentences in the SUSANNE corpus (125,000 words). The gold standard was created by converting the phrase structure annotations for these sentences into dependency notation using the procedure described above. As in the case of WSJ corpus, to account for the differences in the annotation, the dependency relations among words in a noun group were treated as equivalent to each other. So also, dependency relations among words in a verb group were treated as equivalent to each other. There were a total of 126,493 dependency links in the output of the LDA of which, 112,420 also present in the gold standard, resulting in a precision score of 88.8%. There were a total of 140,280 links in the gold standard, resulting a recall of 80.1%. It is interesting to note that although the trigram model was trained on the WSJ corpus, the performance of the LDA on SUSANNE is comparable to the performance of the LDA on the WSJ corpus.

On analyzing the errors, we discovered that several of the errors were due to the approximate nature of the dependency corpus created by the conversion process. A second source of errors was due to the differences in the notion of dependency produced by the LDA system and that resulting from the conversion of the Treebank. This is largely due to a lack of a standard dependency notation.

Corpus	System	# of dependency links	# produced by LDA	# correct	Recall	Precision
Brown	LDA	140,280	126,493	112,420	80.1%	88.8%
WSJ	LDA	47,333	41,009	38,480	82.3%	93.8%

Table 6: Comparative Evaluation of LDA on Wall Street Journal and Brown Corpus

Experiment 2

In this experiment, we used the correct derivation structure produced by XTAG for 1350 WSJ sentences[5] as the gold standard. These sentences were supertagged using the supertagger trained on 8000 sentences of WSJ and dependency annotated using the LDA system. Table 7 shows the performance of the system. Although, the derivation trees produced by the XTAG system are closest in terms of conventions used in the dependency output of the LDA system; there are a few points of divergences, a major one being the annotation for sentence complement verbs. While in the LDA system, the embedded verb depends on the matrix verb the reverse is the case in an LTAG derivation structure. Also, since the XTAG system, as it is implemented does not permit two auxiliary trees to be adjoined at the same node, certain valid derivation structures are not possible in XTAG. A precise formulation of possible derivation structures in LTAG framework is presented in [Schabes and Shieber, 1992].

Table 8 tabulates the percentage of sentences that have zero, one, two and three dependency link errors. In contrast to evaluation against a skeletally bracketed treebank, evaluation against LTAG derivation trees is much more strict. For example, an LTAG derivation contains detailed annotation in terms of the internal structure of the nominal modifiers in Noun Phrases and verbal modifiers in Verb groups. Also, a derivation structure has

[5] sentences of length less than 16

Training Size (words)	Test Size (words)	Recall	Precision
200,000	12,000	83.6%	83.5%

Table 7: Performance of LDA system compared against the XTAG derivation structures

% sentences with 0 errors	% sentences with with $\leq$1 error	% sentences with with $\leq$2 errors	%sentences with with $\leq$3 errors
35%	60.3%	78%	89.8%

Table 8: The percentage of sentences with zero, one, two and three dependency link errors.

a stricter imposition of the argument-adjunct distinction and distinguishes readings that have similar phrase structure trees such as predicative and equative readings and idiomatic and non-idiom readings of a sentence. Further, the derivation structure is much more closer to semantic interpretation of a sentence than the phrase structure. Hence, the performance figures that are shown in Table 7 and Table 8 are more strict and hence more significant than the crossing bracket, precision and recall figures measured against skeletally bracketed corpora.

7 New Models for Supertag Disambiguation

7.1 Head Trigram Model

As is well known, a trigram model is inadequate at modeling dependencies that appear beyond a three word window. One method of allowing supertags beyond the trigram window to influence the selection of a supertag is to allow it to be "visible" in the current context. This can be achieved in one of two ways. First, the size of the window could be increased to be more than three words. One major drawback of this method is that since an a priori bound is set on the size of the window, no matter how large it is, dependencies that appear beyond that window size cannot be adequately modeled.[6] Also, by increasing the size of the window, the number of parameters to be estimated increases exponentially and sparseness of data becomes a significant problem.

An alternate approach to making head supertags "visible" in the current context is to first identify the head positions and then percolate the supertags at the head positions through the preceding context. Once the positions of the head supertags are identified, the contextual probability in a trigram model is changed so as to be conditioned on the supertags of the two previous head words instead of the two immediately preceding previous supertags. Thus, the contextual probability for the head trigram model is given as:

(10) $\Pr(T_1,T_2,\ldots,T_N) \approx \prod_{i=1}^{N} \Pr(T_i \mid T_{H_{i-2}}, T_{H_{i-1}})$
where
$T_{H_{i-1}}$ and $T_{H_{i-2}}$ are the supertags associated with the two previous head words.

The head positions are percolated according to the following equations.

(11)
Initialize:	(H_{-2},H_{-1})	$= (-2,-1)$
Update :	(H_{i-1},H_i)	$= (H_{i-2},H_{i-1})$ if W_i is not a head word
(head positions at i+1)		$= (H_{i-1},i)$ if W_i is a head word

The head trigram supertagger has two passes. In the first pass, the words are annotated for heads and in the second pass the head information is used to compute the appropriate contextual probability. For the first pass, the head words can be identified using a stochastic trigram tagger that uses two labels and tags words either as head words or as non-head words. The training and test material for the head-word tagger was created from the Penn Treebank parses of WSJ sentences using head percolation rules presented in [Jelinek et al., 1994, Magerman, 1995]. The tagger was trained on 1,000,000 words of WSJ corpus and tested on 47,000 words from Section 20 of WSJ. The tagger assigned the correct head-tag for 91.2% of the words, improving on the baseline of 81.4%. Most of head-word tagging errors were mistakes of tagging modifier nouns as head words in nouns group sequences. Using the head information provided by head-word tagger, the head trigram supertagger, trained

[6] An interesting approach of incorporating variable length n-grams in a part-of-speech tagging model is presented in [T.R. Niesler and P.C. Woodland, 1996].

on the 1,000,000 words of WSJ corpus and tested on 47,000 words, assigned the correct supertag for 87% of words. We suspect that since local constraints are not modeled by the head trigram model, the performance of the trigram model is better than this model. We are working towards integrating the two models to produce a mixed model that takes into account both local and non-local constraints.

7.2 Head Trigram Model with Supertags as Feature Vectors

As previously mentioned, supertags contain subcategorization information and localize filler-gap dependency information within the same structure. As a result, for example, there are not only supertags that realize the complements of a transitive verb in their canonical position, but there are supertags that realize complements in passives, supertags for wh-extraction and relative clause for each of the argument positions. These various supertags are present for each subcategorization frame and are transformationally related to one another in a movement-based approach. The supertags related to one another are said to belong to a "family". The supertags for verbs can be organized in a hierarchy as shown in Figure 3. It must be noted that just as the hierarchy in Figure 3 is organized in terms of the subcategorization information, similar hierarchies can be organized based on transformations such as Relative Clause or Wh-extraction. The supertags for other categories such as nouns, adjectives and prepositions can also be organized in similar hierarchies based on features such as modifiers, complements and predicates.

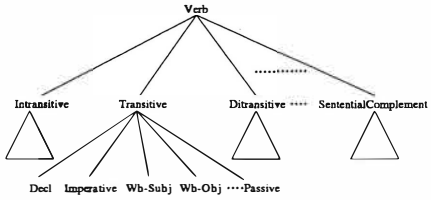

Figure 3: Supertag hierarchy

The representation of supertags as labels, as was done in the previous models, does not exploit the relationship among supertags. Instead, supertags can be viewed as feature vectors where each feature represents a dimension along which the supertags can be clustered. We are currently working on implementing a trigram model with appropriate back-off strategies. A more appropriate model, we believe, for such a distributed encoding of supertags is a model that has the flexibility to select the best back-off strategy as in maximum entropy model. We plan to implement a maximum entropy model for supertagging by the final version of this paper.

8 Conclusions

We have presented the trigram model for supertagging in conjunction with a lightweight dependency analyzer as a robust and efficient partial parser. One of the goals of this work is to explore the limits of the trigram tagging approach and to see how much supertag disambiguation can be done without really parsing. A significant result of the current work is that a trigram model can achieve 92% accuracy for supertagging. Second, we have shown how the disambiguated supertags can be used for partial parsing, detecting noun chunks, verb chunks, preposition phrase attachment, appositives and parenthetical constructions. Using supertags, we have achieved a recall rate of 93.0% and a precision rate of 91.8% for noun chunking, improving on the best known result for noun chunking.

References

[Abney, 1991] Abney, S. (1991). Parsing by chunks. In Berwick, R., Abney, S., and Tenny, C., editors, *Principle-based parsing*. Kluwer Academic Publishers.

[Brill, 1993] Brill, E. (1993). Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio.

[Brill and Resnik, 1994] Brill, E. and Resnik, P. (1994). A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan.

[Church, 1988] Church, K. W. (1988). A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *2nd Applied Natural Language Processing Conference*, Austin, Texas.

[Collins and Brook, 1995] Collins, M. and Brook, J. (1995). Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora*, MIT, Cambridge, Boston.

[Doran, 1996] Doran, C. (1996). Punctuation in Quoted Speech. In *Proceedings of the SIGPARSE96*, Santa Cruz, California.

[Doran et al., 1994] Doran, C., Egedi, D., Hockey, B. A., Srinivas, B., and Zaidel, M. (1994). XTAG System - A Wide Coverage Grammar for English. In *Proceedings of the* 17^{th} *International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan.

[Good, 1953] Good, I. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika 40 (3 and 4)*.

[Gross, 1984] Gross, M. (1984). Lexicon-Grammar and the Syntactic Analysis of French. In *Proceedings of the* 10^{th} *International Conference on Computational Linguistics (COLING'84)*, Stanford, California.

[Harrison et al., 1991] Harrison, P., Abney, S., Fleckenger, D., Gdaniec, C., Grishman, R., Hindle, D., Ingria, B., Marcus, M., Santorini, B., and Strzalkowski, T. (1991). Evaluating syntax performance of parser/grammars of English. In *Proceedings of the Workshop on Evaluating Natural Language Processing Systems, ACL*.

[Hindle and Rooth, 1991] Hindle, D. and Rooth, M. (1991). Structural ambiguity and lexical relations. In *29th Meeting of the Association for Computational Linguistics*, Berkeley, CA.

[Jelinek et al., 1994] Jelinek, F., Lafferty, J., Magerman, D. M., Mercer, R., Ratnaparkhi, A., and Roukos, S. (1994). Decision Tree Parsing using a Hidden Derivation Model. In *Proceedings from the ARPA Workshop on Human Language Technology Workshop*.

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

[Joshi and Srinivas, 1994] Joshi, A. K. and Srinivas, B. (1994). 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.

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

[Marcus et al., 1993] Marcus, M. M., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19.2:313–330.

[Pollard and Sag, 1987] Pollard, C. and Sag, I. A. (1987). *Information-Based Syntax and Semantics. Vol 1: Fundamentals*. CSLI.

[Ramshaw and Marcus, 1995] Ramshaw, L. and Marcus, M. P. (1995). Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora*, MIT, Cambridge, Boston.

[Ratnaparkhi et al., 1994] Ratnaparkhi, A., Reynar, J., and Roukos, S. (1994). A maximum entropy model for prepositional phrase attachment. In *Proceedings of ARPA Workshop on Human Language Technology*, Plainsboro, NJ.

[Sampson, 1994] Sampson, G. (1994). SUSANNE: a Doomsday book of English Grammar. In *Corpus-based Research into Language*. Rodopi, Amsterdam.

[Schabes et al., 1988] Schabes, Y., Abeillé, A., and Joshi, A. K. (1988). Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary.

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

[Sleator and Temperley, 1991] Sleator, D. and Temperley, D. (1991). Parsing English with a Link Grammar. *Technical report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University*.

[Srinivas, 1997] Srinivas, B. (1997). Complexity of Lexical Descriptions and its Relevance to Partial Parsing. *PhD Dissertation, University of Pennsylvania*.

[Steedman, 1987] Steedman, M. (1987). Combinatory Grammars and Parasitic Gaps. *Natural Language and Linguistic Theory*, 5:403–439.

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

[Weischedel et al., 1993] Weischedel, R., Schwartz, R., Palmucci, J., Meteer, M., and Ramshaw, L. (1993). Comping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19.2:359–382.