New Models for Improving Supertag Disambiguation

John Chen*
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716
jchen@cis.udel.edu

Srinivas Bangalore
AT&T Labs Research
180 Park Avenue
P.O. Box 971
Florham Park, NJ 07932
srini@research.att.com

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

Abstract

In previous work, supertag disambiguation has been presented as a robust partial parsing technique. In this paper we present two approaches: contextual models, which exploit a variety of features in order to improve supertag performance, and class-based models, which assign sets of supertags to words in order to substantially improve accuracy with only a slight increase in ambiguity.

1 Introduction

Many natural language applications are beginning to exploit some underlying structure of the language. Roukos (1996) and Jurafsky et al. (1995) use structure-based language models in the context of speech applications. Grishman (1995) and Hobbs et al. (1995) use phrasal information in information extraction. Alshawi (1996) uses dependency information in a machine translation system. The need to impose structure leads to the need to have robust parsers. There have been two main robust parsing paradigms: Finite State Grammar-based approaches (such as Abney (1990), Grishman (1995), and Hobbs et al. (1997)) and Statistical Parsing (such as Charniak (1996), Magerman (1995), and Collins (1996)).

Srinivas (1997a) has presented a different approach called *supertagging* that integrates 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. Supertag disambiguation is resolved by using statistical distributions of supertag co-occurrences collected from a corpus of parses. It results in a representation that is effectively a parse (*almost parse*).

Supertagging has been found useful for a number of applications. For instance, it can be used to speed up conventional chart parsers because it reduces the ambiguity which a parser must face, as described in Srinivas (1997a). Chandrasekhar and Srinivas (1997) has shown that supertagging may be employed in information retrieval. Furthermore, given a sentence aligned parallel corpus of two languages and almost parse information for the sentences of one of the languages, one can rapidly develop a grammar for the other language using supertagging, as suggested by Bangalore (1998).

In contrast to the aforementioned work in supertag disambiguation, where the objective was to provide a direct comparison between trigram models for part-of-speech tagging and supertagging, in this paper our goal is to improve the performance of supertagging using local techniques which avoid full parsing. These supertag disambiguation models can be grouped into *contextual models* and *class based models*. Contextual models use different features in frameworks that exploit the information those features provide in order to achieve higher accuracies in supertagging. For class based models, supertags are first grouped into clusters and words are tagged with clusters of supertags. We develop several automated clustering techniques. We then demonstrate that with a slight increase in supertag ambiguity that supertagging accuracy can be substantially improved.

The layout of the paper is as follows. In Section 2, we briefly review the task of supertagging and the results from previous work. In Section 3, we explore contextual models. In Section 4, we outline various class based approaches. Ideas for future work are presented in Section 5. Lastly, we

*Supported by NSF grants #SBR-9710411 and #GER-9354869

present our conclusions in Section 6.

2 Supertagging

Supertags, the primary elements of the LTAG formalism, attempt to localize dependencies, including long distance dependencies. This is accomplished by grouping syntactically or semantically dependent elements to be within the same structure. Thus, as seen in Figure 1, supertags contain more information than standard part-of-speech tags, and there are many more supertags per word than part-of-speech tags. In fact, supertag disambiguation may be characterized as providing an almost parse, as shown in the bottom part of Figure 1.

Local statistical information, in the form of a trigram model based on the distribution of supertags in an LTAG parsed corpus, can be used to choose the most appropriate supertag for any given word. Joshi and Srinivas (1994) define *supertagging* as the process of assigning the best supertag to each word. Srinivas (1997b) and Srinivas (1997a) have tested the performance of a trigram model, typically used for part-of-speech tagging on supertagging, on restricted domains such as ATIS and less restricted domains such as Wall Street Journal (WSJ).

In this work, we explore a variety of local techniques in order to improve the performance of supertagging. All of the models presented here perform smoothing using a Good-Turing discounting technique with Katz's backoff model. With exceptions where noted, our models were trained on one million words of Wall Street Journal data and tested on 48K words. The data and evaluation procedure are similar to that used in Srinivas (1997b). The data was derived by mapping structural information from the Penn Treebank WSJ corpus into supertags from the XTAG grammar (The XTAG-Group (1995)) using heuristics (Srinivas (1997a)). Using this data, the trigram model for supertagging achieves an accuracy of 91.37%, meaning that 91.37% of the words in the test corpus were assigned the correct supertag.[1]

3 Contextual Models

As noted in Srinivas (1997b), a trigram model often fails to capture the cooccurrence dependencies between a head and its dependents—dependents which might not appear within a trigram's window size. For example, in the sentence "Many Indians *feared* their country *might* split again" the presence of *might* influences the choice of the supertag for *feared*, an influence that is not accounted for by the trigram model. As described below, we show that the introduction of features which take into account aspects of head-dependency relationships improves the accuracy of supertagging.

3.1 One Pass Head Trigram Model

In a head model, the prediction of the current supertag is conditioned not on the immediately preceding two supertags, but on the supertags for the two previous *head* words. This model may thus be considered to be using a context of variable length.[2] The sentence "Many Indians *feared* their country *might* split again" shows a head model's strengths over the trigram model. There are at least two frequently assigned supertags for the word *feared*: a more frequent one corresponding to a subcategorization of NP object (as α_{11} of Figure 1) and a less frequent one to a S complement. The supertag for the word *might*, highly probable to be modeled as an auxiliary verb in this case, provides strong evidence for the latter. Notice that *might* and *feared* appear within a head model's two head window, but not within the trigram model's two word window. We may therefore expect that a head model would make a more accurate prediction.

Srinivas (1997b) presents a *two pass head trigram model*. In the first pass, it tags words as either head words or non-head words. Training data for this pass is obtained using a head percolation table (Magerman (1995)) on bracketed Penn Treebank sentences. After training, head tagging is performed according to Equation 1, where $\tilde{p}$ is the estimated probability and $H(i)$ is a characteristic function which is true iff word i is a head word.

$$H \approx argmax_H \prod_{i=1}^{n} \tilde{p}(w_i|H(i))\tilde{p}(H(i)|H(i-1)H(i-2)) \quad (1)$$

The second pass then takes the words with this head information and supertags them according to Equation 2, where $t_{H(i,j)}$ is the supertag of the

[1] The supertagging accuracy of 92.2% reported in Srinivas (1997b) was based on a different supertag tagset; specifically, the supertag corpus was reannotated with detailed supertags for punctuation and with a different analysis for subordinating conjunctions.

[2] Part of speech tagging models have not used heads in this manner to achieve variable length contexts. Variable length n-gram models, one of which is described in Niesler and Woodland (1996), have been used instead.

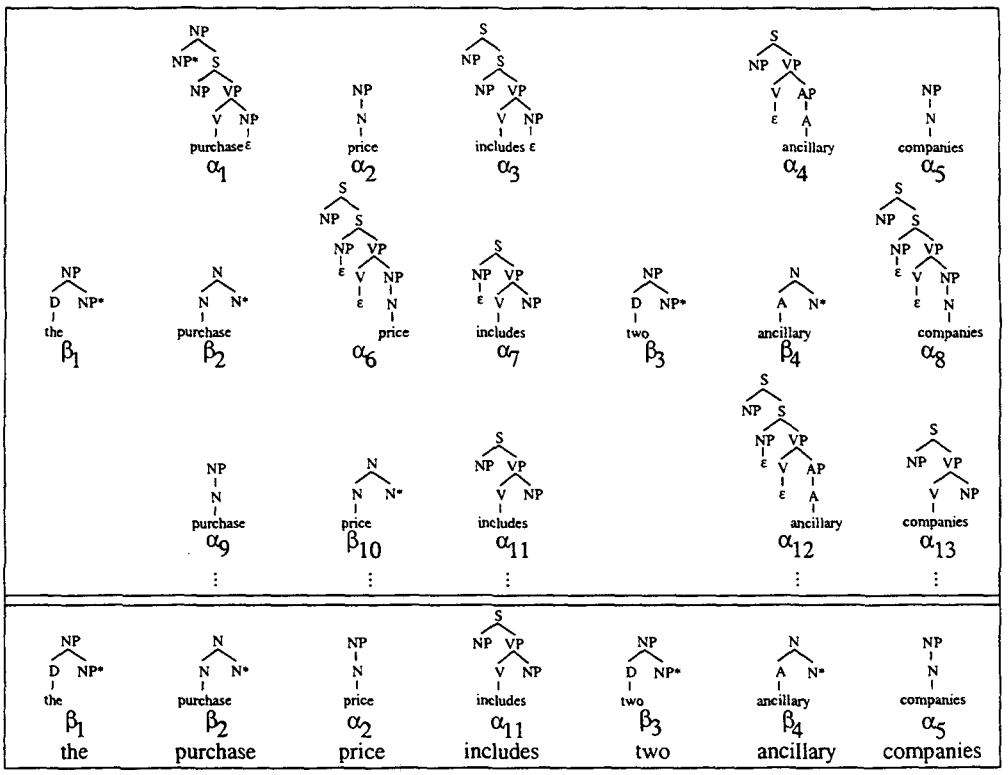

Figure 1: A selection of the supertags associated with each word of the sentence: *the purchase price includes two ancillary companies*

jth head from word i.

$$T \approx argmax_T \prod_{i=1}^{n} \tilde{p}(w_i|t_i)\tilde{p}(t_i|t_{H(i,-1)}t_{H(i,-2)}) \quad (2)$$

This model achieves an accuracy of 87%, lower than the trigram model's accuracy.

Our current approach differs significantly. Instead of having heads be defined through the use of the head percolation table on the Penn Treebank, we define headedness in terms of the supertags themselves. The set of supertags can naturally be partitioned into head and non-head supertags. Head supertags correspond to those that represent a predicate and its arguments, such as α_3 and α_7. Conversely, non-head supertags correspond to those supertags that represent modifiers or adjuncts, such as β_1 and β_2.

Now, the tree that is assigned to a word during supertagging determines whether or not it is to be a head word. Thus, a simple adaptation of the Viterbi algorithm suffices to compute Equation 2 in a single pass, yielding a *one pass head trigram model*. Using the same training and test data, the one pass head model achieved 90.75% accuracy, constituting a 28.8% reduction in error over the two pass head trigram model. This improvement may come from a reduction in error propagation or the richer context that is being used to predict heads.

3.2 Mixed Head and Trigram Models

The head model skips words that it does not consider to be head words and hence may lose valuable information. The lack of immediate local context hurts the head model in many cases, such as selection between head noun and noun modifier, and is a reason for its lower performance relative to the trigram model. Consider the phrase "..., or $ 2.48 a share." The word *2.48* may either be associated with a determiner phrase supertag (β_1) or a noun phrase supertag (α_9). Notice that *2.48* is immediately preceded by *$* which is extremely likely to be supertagged as a determiner phrase (β_1). This is strong evidence that *2.48* should be supertagged as α_9. A pure head model cannot consider this particular fact, however, because β_1 is not a head supertag. Thus, local context and long distance head dependency relationships are both important for accurate supertagging.

A *5-gram mixed model* that includes both the trigram and the head trigram context is one approach to this problem. This model achieves a performance of 91.50%, an improvement over both

Previous Context	Current Supertag	Next Context
$t_{H(i,-2)}t_{H(i,-1)}$	$t_{H(i,0)}$	$t_{H(i,-1)}t_{H(i,0)}$
$t_{H(i,-2)}t_{H(i,-1)}$	$t_{LM(i,0)}$	$t_{H(i,-1)}t_{LM(i,0)}$
$t_{H(i,-2)}t_{H(i,-1)}$	$t_{RM(i,0)}$	$t_{H(i,-2)}t_{H(i,-1)}$
$t_{H(i,-1)}t_{LM(i,-1)}$	$t_{H(i,0)}$	$t_{H(i,-1)}t_{H(i,0)}$
$t_{H(i,-1)}t_{LM(i,-1)}$	$t_{LM(i,0)}$	$t_{H(i,-1)}t_{LM(i,0)}$
$t_{H(i,-1)}t_{LM(i,-1)}$	$t_{RM(i,0)}$	$t_{H(i,-1)}t_{RM(i,0)}$

Table 1: In the 3-gram mixed model, previous conditioning context and the current supertag deterministically establish the next conditioning context. H, LM, and RM denote the entities head, left modifier, and right modifier, respectively.

Model	Context	Accuracy
Trigram	$t_{i-1}t_{i-2}$	91.37
Head Trigram	$t_{H(i,-1)}t_{H(i,-2)}$	90.75
5-gram Mix	$t_{i-1}t_{i-2}$ $t_{H(i,-1)}t_{H(i,-2)}$	91.50
3-gram Mix	$t_{cntxt(i,-1)}t_{cntxt(i,-2)}$	91.79
Head Word	$w_{(i,-1)}w_{(i,-2)}$	88.16
Mix Word	$t_{i-1}t_{i-2}$ $w_{H(i,-1)}w_{H(i,-2)}$	89.46

Table 2: Single classifier contextual models that have been explored along with the contexts they consider and their accuracies

the trigram model and the head trigram model. We hypothesize that the improvement is limited because of a large increase in the number of parameters to be estimated.

As an alternative, we explore a *3-gram mixed model* that incorporates nearly all of the relevant information. This mixed model may be described as follows. Recall that we partition the set of all supertags into heads and modifiers. Modifiers have been defined so as to share the characteristic that each one either modifies exactly one item to the right or one item to the left. Consequently, we further divide modifiers into *left modifiers* (β_4) and *right modifiers*. Now, instead of fixing the conditioning context to be either the two previous tags (as in the trigram model) or the two previous head tags (as in the head trigram model) we allow it to vary according to the identity of the current tag and the previous conditioning context, as shown in Table 1. Intuitively, the mixed model is like the trigram model except that a modifier tag is discarded from the conditioning context when it has found an object of modification. The mixed model achieves an accuracy of 91.79%, a significant improvement over both the head trigram model's and the trigram model's accuracies, $p < 0.05$. Furthermore, this mixed model is computationally more efficient as well as more accurate than the 5-gram model.

3.3 Head Word Models

Rather than head *supertags*, head *words* often seem to be more predictive of dependency relations. Based upon this reflection, we have implemented models where head words have been used as features. The *head word model* predicts the current supertag based on two previous head words (backing off to their supertags) as shown in Equation 3.

$$T \approx argmax_T \prod_{i=1}^{n} \tilde{p}(w_i|t_i)\tilde{p}(t_i|w_{H(i,-1)}w_{H(i,-2)}) \quad (3)$$

The *mixed trigram and head word model* takes into account local (supertag) context and long distance (head word) context. Both of these models appear to suffer from severe sparse data problems. It is not surprising, then, that the head word model achieves an accuracy of only 88.16%, and the mixed trigram and head word model achieves an accuracy of 89.46%. We were only able to train the latter model with 250K of training data because of memory problems that were caused by computing the large parameter space of that model.

The salient characteristics of models that have been discussed in this subsection are summarized in Table 2.

3.4 Classifier Combination

While the features that our new models have considered are useful, an n-gram model that considers all of them would run into severe sparse data problems. This difficulty may be surmounted through the use of more elaborate backoff techniques. On the other hand, we could consider using decision trees at choice points in order to decide which features are most relevant at each point. However, we have currently experimented with *classifier combination* as a means of ameliorating the sparse data problem while making use of the feature combinations that we have introduced.

In this approach, a selection of the discussed models is treated as a different classifier and is trained on the same data. Subsequently, each classifier supertags the test corpus separately. Finally,

	Trigram	Head Trigram	Head Word	3-gram Mix	Mix Word
Trigram	91.37	91.87*	91.65	91.96	91.55
Head Trigram		90.75	90.96	91.95	91.35*
Head Word			88.16	91.88	90.51*
3-gram Mix				91.79	91.87
Mix Word					89.46

Table 3: Accuracies of Single Classifiers and Pairwise Combination of Classifiers.

their predictions are combined using various voting strategies.

The same 1000K word test corpus is used in models of classifier combination as is used in previous models. We created three distinct partitions of this 1000K word corpus, each partition consisting of a 900K word training corpus and a 100K word tune corpus. In this manner, we ended up with a total of 300K word tuning data.

We consider three voting strategies suggested by van Halteren et al. (1998): *equal vote*, where each classifier's vote is weighted equally, *overall accuracy*, where the weight depends on the overall accuracy of a classifier, and *pairwise voting*. Pairwise voting works as follows. First, for each pair of classifiers a and b, the empirical probability $\tilde{p}(t_{correct}|t_{classifier-a}t_{classifier-b})$ is computed from tuning data, where $t_{classifier-a}$ and $t_{classifier-b}$ are classifier a's and classifier b's supertag assignment for a particular word respectively, and $t_{correct}$ is the correct supertag. Subsequently, on the test data, each classifier pair votes, weighted by overall accuracy, for the supertag with the highest empirical probability as determined in the previous step, given each individual classifier's guess.

The results from these voting strategies are positive. Equal vote yields an accuracy of 91.89%. Overall accuracy vote has an accuracy of 91.93%. Pairwise voting yields an accuracy of 92.19%, the highest supertagging accuracy that has been achieved, a 9.5% reduction in error over the trigram model.

The table of accuracy of combinations of pairs of classifiers is shown in Table 3.[3] The efficacy of pairwise combination (which has significantly fewer parameters to estimate) in ameliorating the sparse data problem can be seen clearly. For example, the accuracy of pairwise combination of head classifier and trigram classifier exceeds that of the 5-gram mixed model. It is also marginally, but not significantly, higher than the 3-gram mixed model. It is also notable that the pairwise combination of the head word classifier and the mix word classifier yields a significant improvement over either classifier, $p < 0.05$, considering the disparity between the accuracies of its component classifiers.

3.5 Further Evaluation

We also compare various models' performance on base-NP detection and PP attachment disambiguation. The results will underscore the adroitness of the classifier combination model in using both local and long distance features. They will also show that, depending on the ultimate application, one model may be more appropriate than another model.

A base-NP is a non-recursive NP structure whose detection is useful in many applications, such as information extraction. We extend our supertagging models to perform this task in a fashion similar to that described in Srinivas (1997b). Selected models have been trained on 200K words. Subsequently, after a model has supertagged the test corpus, a procedure detects base-NPs by scanning for appropriate sequences of supertags. Results for base-NP detection are shown in Table 4. Note that the mixed model performs nearly as well as the trigram model. Note also that the head trigram model is outperformed by the other models. We suspect that unlike the trigram model, the head model does not perform the accurate modeling of local context which is important for base-NP detection.

In contrast, information about long distance dependencies are more important for the the PP attachment task. In this task, a model must decide whether a PP attaches at the NP or the VP level. This corresponds to a choice between two PP supertags: one associated with NP attachment, and another associated with VP attachment. The trigram model, head trigram model, 3-gram mixed model, and classifier combination model perform at accuracies of 78.53%, 79.56%, 80.16%, and 82.10%, respectively, on the PP at-

[3]Entries marked with an asterisk ("*") correspond to cases where the pairwise combination of classifiers was significantly better than either of their component classifiers, $p < 0.05$.

	Recall	Precision
Trigram	93.75	93.00
3-gram Mix	93.65	92.63
Head Trigram	91.17	89.72
Classifier Combination	94.00	93.17

Table 4: Some contextual models' results on base-NP chunking

tachment task. As may be expected, the trigram model performs the worst on this task, presumably because it is restricted to considering purely local information.

4 Class Based Models

Contextual models tag each word with the single most appropriate supertag. In many applications, however, it is sufficient to reduce ambiguity to a small number of supertags per word. For example, using traditional TAG parsing methods, such are described in Schabes (1990), it is inefficient to parse with a large LTAG grammar for English such as XTAG (The XTAG-Group (1995)). In these circumstances, a single word may be associated with hundreds of supertags. Reducing ambiguity to some small number k, say $k < 5$ supertags per word[4] would accelerate parsing considerably.[5] As an alternative, once such a reduction in ambiguity has been achieved, partial parsing or other techniques could be employed to identify the best single supertag. These are the aims of *class based models*, which assign a small set of supertags to each word. It is related to work by Brown et al. (1992) where mutual information is used to cluster words into classes for language modeling. In our work with class based models, we have considered only trigram based approaches so far.

4.1 Context Class Model

One reason why the trigram model of supertagging is limited in its accuracy is because it considers only a small contextual window around the word to be supertagged when making its tagging decision. Instead of using this limited context to pinpoint the exact supertag, we postulate that it may be used to predict certain structural characteristics of the correct supertag with much higher accuracy. In the *context class model*, supertags that share the same characteristics are grouped into classes and these classes, rather than individual supertags, are predicted by a trigram model. This is reminiscent of Samuelsson and Reich (1999) where some part of speech tags have been compounded so that each word is deterministically in one class.

The grouping procedure may be described as follows. Recall that each supertag corresponds to a lexicalized tree $t \in G$, where G is a particular LTAG. Using standard FIRST and FOLLOW techniques, we may associate t with FOLLOW and PRECEDE sets, FOLLOW(t) being the set of supertags that can immediately follow t and PRECEDE(t) being those supertags that can immediately precede t. For example, an NP tree such as β_1 would be in the FOLLOW set of a supertag of a verb that subcategorizes for an NP complement. We partition the set of all supertags into classes such that all of the supertags in a particular class are associated with lexicalized trees with the same PRECEDE and FOLLOW sets. For instance, the supertags t_1 and t_2 corresponding respectively to the NP and S subcategorizations of a verb *feared* would be associated with the same class. (Note that a head NP tree would be a member of both FOLLOW(t_1) and FOLLOW(t_2).)

The context class model predicts sets of supertags for words as follows. First, the trigram model supertags each word w_i with supertag t_i that belongs to class C_i.[6] Furthermore, using the training corpus, we obtain set D_i which contains all supertags t such that $\tilde{p}(w_i|t) > 0$. The word w_i is relabeled with the set of supertags $C_i \cap D_i$.

The context class model trades off an increased ambiguity of 1.65 supertags per word on average, for a higher 92.51% accuracy. For the purpose of comparison, we may compare this model against a baseline model that partitions the set of all supertags into classes so that all of the supertags in one class share the same preterminal symbol, i.e., they are anchored by words which share the same part of speech. With classes defined in this manner, call C'_i the set of supertags that belong to the class which is associated with word w_i in the test corpus. We may then associate with word w_i the set of supertags $C'_i \cap D_i$, where D_i is defined as above. This baseline procedure yields an aver-

[4] For example, the n-best model, described below, achieves 98.4% accuracy with on average 4.8 supertags per word.

[5] An alternate approach to TAG parsing that effectively shares the computation associated with each lexicalized elementary tree (supertag) is described in Evans and Weir (1998). It would be worth comparing both approaches.

[6] For class models, we have also experimented with a variant where the classes are assigned to words through the model $C \approx argmax_C \prod_{i=1}^{n} \tilde{p}(w_i|C_i)\tilde{p}(C_i|C_{i-1}C_{i-2})$. In general, we have found this procedure to give slightly worse results.

age ambiguity of 5.64 supertags per word with an accuracy of 97.96%.

4.2 Confusion Class Model

The *confusion class model* partitions supertags into classes according to an alternate procedure. Here, classes are derived from a confusion matrix analysis of errors which the trigram model makes while supertagging. First, the trigram model supertags a tune set. A confusion matrix is constructed, recording the number of times supertag t_i was confused for supertag t_j, or vice versa, in the tune set. Based on the top k pairs of supertags that are most confused, we construct classes of supertags that are confused with one another. For example, let t_1 and t_2 be two PP supertags which modify an NP and VP respectively. The most common kind of mistake that the trigram model made on the tune data was to mistag t_1 as t_2, and vice versa. Hence, t_1 and t_2 are clustered by our method into the same confusion class. The second most common mistake was to confuse supertags that represent verb modifier PPs and those that represent verb argument PPs, while the third most common mistake was to confuse supertags that represent head nouns and noun modifiers. These, too, would form their own classes.

The confusion class model predicts sets of supertags for words in a manner similar to the context class model. Unlike the context class model, however, in this model we have to choose k, the number of pairs of supertags which are extracted from the confusion matrix over which confusion classes are formed. In our experiments, we have found that with $k = 10$, $k = 20$, and $k = 40$, the resulting models attain 94.61% accuracy and 1.86 tags per word, 95.76% accurate and 2.23 tags per word, and 97.03% accurate and 3.38 tags per word, respectively.[7]

Results of these, as well as other models discussed below, are plotted in Figure 2. The *n-best model* is a modification of the trigram model in which the n most probable supertags per word are chosen. The *classifier union* result is obtained by assigning a word w_i a set of supertags $t_{i1}, \ldots, t_{ik}$ where to t_{ij} is the jth classifier's supertag assignment for word w_i, the classifiers being the models discussed in Section 3. It achieves an accuracy of 95.21% with 1.26 supertags per word.

[7]Again, for the class C assign to a given word w_i, we consider only those tags $t_i \in C$ for which $\tilde{p}(w_i|t_i) > 0$.

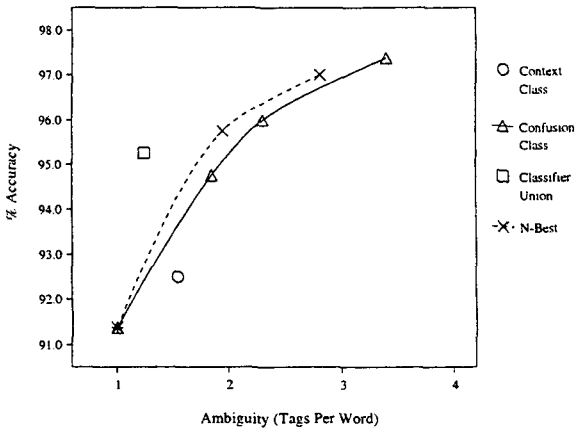

Figure 2: Ambiguity versus Accuracy for Various Class Models

5 Future Work

We are considering extending our work in several directions. Srinivas (1997b) discussed a lightweight dependency analyzer which assigns dependencies assuming that each word has been assigned a unique supertag. We are extending this algorithm to work with class based models which narrows down the number of supertags per word with much higher accuracy. Aside from the n-gram modeling that was a focus of this paper, we would also like to explore using other kinds of models, such as maximum entropy.

6 Conclusions

We have introduced two different kinds of models for the task of supertagging. Contextual models show that features for accurate supertagging only produce improvements when they are appropriately combined. Among these models were: a one pass head model that reduces propagation of head detection errors of previous models by using supertags themselves to identify heads; a mixed model that combines use of local and long distance information; and a classifier combination model that ameliorates the sparse data problem that is worsened by the introduction of many new features. These models achieve better supertagging accuracies than previously obtained. We have also introduced class based models which trade a slight increase in ambiguity for significantly higher accuracy. Different class based methods are discussed, and the tradeoff between accuracy and ambiguity is demonstrated.

References

Steven Abney. 1990. Rapid Incremental parsing

with repair. In *Proceedings of the 6th New OED Conference: Electronic Text Research*, pages 1–9, University of Waterloo, Waterloo, Canada.

Hiyan Alshawi. 1996. Head automata and bilingual tiling: translation with minimal representations. In *Proceedings of the 34th Annual Meeting Association for Computational Linguistics*, Santa Cruz, California.

Srinivas Bangalore. 1998. Transplanting Supertags from English to Spanish. In *Proceedings of the TAG+4 Workshop*, Philadelphia, USA.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language *Computational Linguistics*, 18.4:467-479.

R. Chandrasekhar and B. Srinivas. 1997. Using supertags in document filtering: the effect of increased context on information retrieval In *Proceedings of Recent Advances in NLP '97.*

Eugene Charniak. 1996. Tree-bank Grammars. Technical Report CS-96-02, Brown University, Providence, RI.

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

Roger Evans and David Weir. 1998. A Structure-sharing Parser for Lexicalized Grammars. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, Montreal.

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

H. van Halteren, J. Zavrel, and W. Daelmans. 1998. Improving Data Driven Wordclass Tagging by System Combination. In *Proceedings of COLING-ACL 98*, Montreal.

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

Jerry R. Hobbs, 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 Schabes Y., editors, *Finite State Devices for Natural Language Processing.* MIT Press, Cambridge, Massachusetts.

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

D. Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary Tajchman, and Nelson Morgan. 1995. Using a Stochastic CFG as a Language Model for Speech Recognition. In *Proceedings, IEEE ICASSP*, Detroit, Michigan.

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

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

S. Roukos. 1996. Phrase structure language models. In *Proc. ICSLP '96*, volume supplement, Philadelphia, PA, October.

Christer Samuelsson and Wolfgang Reich. 1999. A Class-based Language Model for Large Vocabulary Speech Recognition Extracted from Part-of-Speech Statistics. In *Proceedings, IEEE ICASSP.*

Yves Schabes. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars.* Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

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

B. Srinivas. 1997b. Performance Evaluation of Supertagging for Partial Parsing. In *Proceedings of Fifth International Workshop on Parsing Technology*, Boston, USA, September.

R. Weischedel., R. Schwartz, J. Palmucci, M. Meteer, and L. Ramshaw. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19.2:359-382.

The XTAG-Group. 1995. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 95-03, University of Pennsylvania, Philadelphia, PA.