Coping with Ambiguity and Unknown Words through Probabilistic Models

Ralph Weischedel*
BBN Systems and Technologies

Marie Meteer†
Rensselaer Polytechnic Institute

Richard Schwartz*
BBN Systems and Technologies

Lance Ramshaw‡
Bowdoin College

Jeff Palmucci*
BBN Systems and Technologies

From spring 1990 through fall 1991, we performed a battery of small experiments to test the effectiveness of supplementing knowledge-based techniques with probabilistic models. This paper reports our experiments in predicting parts of speech of highly ambiguous words, predicting the intended interpretation of an utterance when more than one interpretation satisfies all known syntactic and semantic constraints, and learning case frame information for verbs from example uses.

From these experiments, we are convinced that probabilistic models based on annotated corpora can effectively reduce the ambiguity in processing text and can be used to acquire lexical information from a corpus, by supplementing knowledge-based techniques.

Based on the results of those experiments, we have constructed a new natural language system (PLUM) for extracting data from text, e.g., newswire text.

1. Introduction

Natural language processing, and AI in general, have focused mainly on building rule-based systems with carefully handcrafted rules and domain knowledge. Our own natural language database query systems, JANUS (Weischedel et al. 1989), Parlance^{TM},[1] and Delphi (Stallard 1989), have used these techniques quite successfully. However, as we move from the application of understanding database queries in limited domains to applications of processing open-ended text, we found challenges that questioned our previous assumptions and suggested probabilistic models instead.

1. We could no longer assume a limited vocabulary. Rather in the domain of terrorist incidents of the Third Message Understanding conference (MUC-3) (Sundheim 1991), roughly 20,000 vocabulary items appear in a corpus 430,000 words long. Additional text from that domain would undoubtedly contain new words. *Probabilistic models offer a mathematically grounded, empirically based means of predicting the most likely interpretation.*

* BBN Systems and Technologies, 70 Fawcett Street, Cambridge MA 02138.
† Sage Lab, Rensselaer Polytechnic Institute, Troy NY 12180.
‡ Computer Science Department, Bowdoin College, Brunswick ME 04011.
1 Parlance is a trademark of BBN Systems and Technologies.

2. Having semantics for all (or even most) of the words of the vocabulary would violate the limited domain assumption, since roughly 50% of the message stream mentions no terrorist incident, and even those that do may be primarily about a different topic or topics. Therefore, the power of semantic constraints in limited domains would be diluted. *Probability models could be employed where less knowledge was available.*
3. Given the vocabulary size, we could not expect to give full syntactic or semantic features. The labor for handcrafted definitions would not be warranted. *Statistical language models have a learning component that might supplement handcrafted knowledge.*
4. Purely rule-based techniques seemed too brittle for dealing with the variety of constructions, the long sentences (averaging 29 words per sentence), and the degree of unexpected input. *Statistical models based on local information* (e.g., DeRose 1988; Church 1988) *might operate effectively in spite of sentence length and unexpected input.*

To see whether our four hypotheses (in italics above) effectively addressed the four concerns above, we chose to test the hypotheses on two well-known problems: ambiguity (both at the structural level and at the part-of-speech level) and inferring syntactic and semantic information about unknown words.

Guided by the past success of probabilistic models in speech processing, we have integrated probabilistic models into our language processing systems. Early speech research used purely knowledge-based approaches, analogous to knowledge-based approaches in NLP systems today. These required much detailed, handcrafted knowledge from several sources (e.g., acoustic and phonetic). However, when it became clear that these techniques were too brittle and not scalable, speech researchers turned to probabilistic models. These provided a flexible control structure for combining multiple sources of knowledge (providing improved accuracy and ability to deal with more complex domains) and algorithms for training the system on large bodies of data (providing reduced cost in moving the technology to a new application domain).

Since probability theory offers a general mathematical modeling tool for estimating how likely an event is, probability theory may be applied at all levels in natural language processing, because some set of events can be associated with each algorithm. For example, in morphological processing in English (Section 2), the events are the use of a word with a particular part of speech in a string of words. At the level of syntax (Section 3), an event is the use of a particular structure; the model predicts what the most likely rule is given a particular situation. One can similarly use probabilities for assigning semantic structure (Section 4).

We report in Section 2 on our experiments on the assignment of part of speech to words in text. The effectiveness of such models is well known (DeRose 1988; Church 1988; Kupiec 1989; Jelinek 1985), and they are currently in use in parsers (e.g. de Marcken 1990). Our work is an incremental improvement on these models in three ways: (1) Much less training data than theoretically required proved adequate; (2) we integrated a probabilistic model of word features to handle unknown words uniformly within the probabilistic model and measured its contribution; and (3) we have applied the forward–backward algorithm to accurately compute the most likely tag set.

In Section 3, we demonstrate that probability models can improve the performance of knowledge-based syntactic and semantic processing in dealing with structural ambiguity and with unknown words. Though the probability model employed is not new, our empirical findings are novel. When a choice among alternative interpretations produced by a unification-based parser and semantic interpreter must be made, a simple context-free probability model reduced the error rate by a factor of two compared with using no model. It is well known that a unification parser can process an unknown word by collecting the assumptions it makes while trying to find an interpretation for a sentence. As a second result, we found that adding a context-free probability model improved the unification predictions of syntactic and semantic properties of an unknown word, reducing the error rate by a factor of two compared with no model.

In Section 4, we report an experiment in learning case frame information of unknown verbs from examples. The probabilistic algorithm is critical to selecting the appropriate generalizations to make from a set of examples. The effectiveness of the semantic case frames inferred is measured by testing how well those case frames predict the correct attachment point for prepositional phrases. In this case, a significant new model synthesizing both semantic and syntactic knowledge is employed.

2. POST: Using Probabilities to Tag Part of Speech

Identifying the part of speech of a word illustrates both the problem of ambiguity and the problem of unknown words. Many words are ambiguous in several ways, as in the following:

a *round* table: adjective

a *round* of cheese: noun

to *round* out your interests: verb

to work the year *round*: adverb

Even in context, part of speech can be ambiguous, as in the famous example: "Time flies like an arrow," where the first three words are ambiguous in two ways, resulting in four grammatical interpretations of the sentence. In processing text such as newswire, ambiguity at the word level is high. In an analysis of texts from the *Wall Street Journal* (*WSJ*), we found that the average number of parts of speech per word was approximately two.

Determining the part of speech of an unknown word can help the system to know how the word functions in the sentence; for instance, that it is a verb stating an action or state of affairs, that it is a common noun stating a class of persons, places, or things, that it is a proper noun naming a particular person, place, or thing, etc. If it can do that well, then more precise classification and understanding is feasible.

The most critical feature to us is to have local criteria for ranking the alternative parts of speech, rather than relying solely on a globally correct parse. The probability model we selected offers these features.

The name of our component for part of speech is POST (part-of-speech tagger).

2.1 The n-Gram Model

In our work, we have used well-known probability models known as Hidden Markov Models; therefore, none of the background in Section 2.1 is novel. If we want to determine the most likely syntactic part of speech or *tag* for each word in a sentence, we can formulate a probabilistic tagging model. Let us assume that we want to know the most likely tag sequence, $T = \{t_1, t_2, \ldots, t_n\}$, given a particular word sequence,

$W = \{w_1, w_2, \dots w_n\}$. Using Bayes' rule we know that

$$p(T \mid W) = \frac{p(T)p(W \mid T)}{p(W)}$$

where $p(T)$ is the a priori probability of tag sequence T, $p(W \mid T)$ is the conditional probability of word sequence W occurring given that a sequence of tags T occurred, and $p(W)$ is the unconditioned probability of word sequence W. Then, in principle, we can consider all possible tag sequences, evaluate $p(T \mid W)$ of each, and choose the tag sequence T that is most likely, i.e., the sequence that maximizes $p(T \mid W)$. Since W is the same for all hypothesized tag sequences, we can disregard $p(W)$.

We can rewrite the probability of each sequence as a product of the conditional probabilities of each word or tag given all of the previous tags.

$$p(T \mid W)p(W) = \prod_{i=1}^{n} p(t_i \mid t_{i-1}, t_{i-2}, \dots, t_1)p(w_i \mid t_i \dots t_1, w_{i-1} \dots w_1)$$

Typically, one makes two simplifying assumptions to cut down on the number of probabilities to be estimated. Rather than assuming w_i depends on all previous words and all previous tags, one assumes w_i depends only on t_i. This independence assumption, of course, is not correct. Yet, it so reduces the number of probabilities that must be estimated and therefore so reduces the amount of data needed to estimate probabilities, that it is a worthwhile simplifying assumption. It is an empirical issue whether alternative assumptions would yield significantly better performance.

Second, rather than assuming the tag t_i depends on the full sequence of previous tags, we can assume that local context is sufficient. Typically, individuals have assumed tag t_i depends only on t_{i-1} and t_{i-2} (a tri-tag model) or only on t_{i-1} (a bi-tag model). This assumed locality is termed a Markov independence assumption.

Using a tri-tag model, we then have the following:

$$p(T \mid W)p(W) = p(t_1)p(t_2 \mid t_1)\prod_{i=3}^{n} p(t_i \mid t_{i-1}, t_{i-2}) \left[\prod_{i=1}^{n} p(w_i \mid t_i)\right]$$

If we have sufficient training data, we can estimate the tag n-gram sequence of probabilities and the probability of each word given a tag (lexical probabilities). Using a tagged corpus to train the model is called "supervised training," since a human has prepared the correct training data. We conducted supervised training to derive both a *bi-tag* and a *tri-tag model* based on a corpus from the University of Pennsylvania, which was created as part of the TREEBANK project (Santorini 1990) consisting of *Wall Street Journal* (*WSJ*) articles, texts from the Library of America, transcribed radio broadcasts, and transcribed dialogues. The full TREEBANK consists of approximately 4 million words of text. Of the 47 parts of speech, 36 are word tags, and 11 are punctuation tags. Of the word tags, 22 are tags for open class words and 14 for closed class words. Each word or punctuation mark has been tagged, as shown in the following example, where NNS is plural noun; VBD is past tense verb; RB is adverbial; VBN is past participle verb.

Terms/NNS *were*/VBD *not*/RB *disclosed*/VBN . / .

A bi-tag model predicts the relative likelihood of a particular tag given the preceding tag, e.g., how likely is the tag VBD on the second word in the above example, given that the previous word was tagged NNS. A tri-tag model predicts the relative likelihood of a particular tag given the two preceding tags, e.g., how likely is the tag RB on the third word in the above example, given that the two previous words were tagged NNS and VBD. While the bi-tag model is faster at processing time, the tri-tag model has a lower error rate.

The algorithm for supervised training is straightforward. One counts for each possible pair of tags, the number of times that the pair was followed by each possible third tag. The number of times a given third tag t' occurs after tags t_1 and t_2 divided by the number of times t_1 and t_2 are followed by any third tag is an estimate of the probability of $p(t' \mid t_2, t_1)$.

One also estimates from the training data the conditional probability of each particular word given a known tag (e.g., how likely is the word "terms" if the tag is NNS); this is called the "word emit" probability. This is simply the number of times a particular word appears as part of speech t, divided by the number of times part of speech t appears in the corpus.

No matter how large the training corpus, one may not see all pairs or triples of tags, nor all words used in each part of speech possible in the language, nor all words. It seems unwise to assume that the probability of an unseen event is zero. To deal with the previously unseen, one employs one of several estimation techniques called "padding." Thus far, we have employed the simplest of these techniques for estimating $p(t_3 \mid t_2 t_1)$ if $t_1 t_2 t_3$ was not present in the training corpus. Suppose triples beginning with $t_1 t_2$ appear m times in the corpus. Suppose further that for j distinct tags t, $t_1 t_2 t'$ was not present in the corpus. Then, we estimate $p(t \mid t_2 t_1) = 1/m$ (as if it actually had been seen once). So that the probability of tags given $t_1 t_2$ sum to one, we subtract $1/jm$ from the probability of each triple that actually was observed in the corpus, i.e., if $t_1 t_2 t'$ was observed k times in the corpus, then we estimate $p(t' \mid t_1 t_2) = k/m - 1/jm$.

Given these probabilities, one can then find the most likely tag sequence for a given word sequence. Using the Viterbi algorithm, we selected the path whose overall probability was highest, and then took the tag predictions from that path. We replicated the earlier results that this process is able to predict the parts of speech with only a 3–4% error rate when the possible parts of speech of each of the words in the corpus are known. This is in fact about the rate of discrepancies among human taggers on the TREEBANK project (Marcus, Santorini, and Magerman 1990).

2.2 How Much Training Data Is Required?

While supervised training is shown here to be very effective, it requires a correctly tagged corpus. How much manually annotated data is required?

In our experiments, we demonstrated that the training set can, in fact, be much smaller than might have been expected. One rule of thumb suggests that the training set needs to be large enough to contain on average ten instances of each type of tag sequence that occurs. This would imply that a tri-tag model using 47 possible parts of speech would need a bit more than 1 million words of training, if all possible tag sequences occur. However, we found that much less training data is necessary, since many possible sequences do not occur.

It can be shown that if the average number of tokens of each tri-gram that has been observed is ten, then the lower bound on the probability of new tri-grams is $1/10$. Thus the likelihood of a new tri-gram is fairly low.

While theoretically the set of possible events is all permutations of the tags, in practice only a relatively small number of tri-tag sequences actually occur. Out of about 97,000 possible triples, we found only 6,170 unique triples when we trained on 64,000 words, and about 10,000 when we trained on 1,000,000 words. Thus, even

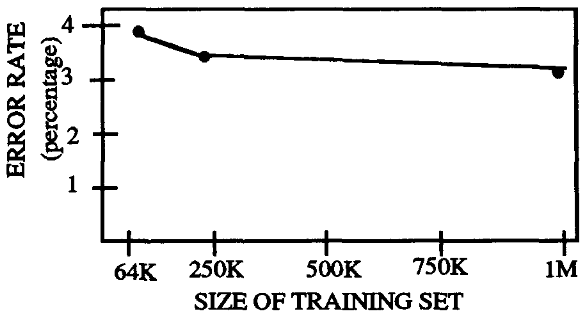


Figure 1
Size of tri-tag training sets.

though an additional 4,000 sequences are observed in the full training set, they are so rare (0.4%) that they do not significantly affect the overall accuracy.

In our initial experiments, which were limited to known words, the error rate for a supervised tri-tag model increased only from 3.30% to 3.87% when the size of the training set was reduced from 1 million words to 64,000 words (see Figure 1). All that is really necessary, recalling the rule of thumb, is enough training to allow for ten of each of the tag sequences that do occur.

This result is applicable to new tag sets, subdomains, or languages. We simply continue to increase the amount of training data until the number of training tokens is at least ten times the number of different sequences observed so far. Alternatively, we can stop when the singleton events account for a small enough percentage (say 5%) of the total data. Thus, in applications such as tagging, where a significant number of the theoretically possible events do not occur in practice, we can use supervised training of probabilistic models without needing prohibitively large corpora.

Of course, performance of POST is also affected by the estimates of $p(w_i \mid t_i)$ for known words and unknown words. How to estimate $p(w_i \mid t_i)$ for unknown words is covered in the next section. For an observed word, a small training set of 64,000 words may still be adequate for estimates of $p(w_i \mid t_i)$. We found that by treating words observed only once as if they had not been observed at all (and are thus handled by the probabilistic models for unknown words) that performance actually increased slightly. This suggests that adequate performance can be obtained from a relatively small training set.

We are not aware of any other published studies documenting empirically the impact of training set size on performance.

2.3 Unknown Words

Sources of open-ended text, such as a newswire, present natural language processing technology with a major challenge: what to do with words the system has never seen before. Current technology depends on handcrafted linguistic and domain knowledge. For instance, the system that performed most successfully in the evaluation of software to extract data from text at the Second Message Understanding Conference held at the Naval Ocean Systems Center, June 1989, would simply halt processing a sentence when a new word was encountered.

Using the UPenn set of parts of speech, unknown words can be in any of 22 categories. A tri-tag model can be used to estimate the most probable one. Random choice among the 22 open classes would be expected to show an error rate for new words of 95%. The best previously reported error rate based on probabilistic models was 75% (Kuhn and DeMori 1990).

In our first tests using the tri-tag model we showed an error rate of only 51.6%. However, this model only took into account the context of the word, and no information about the word itself. In many languages, including English, word endings give strong indicators of the part of speech. Furthermore, capitalization information, when available, can help to indicate whether a word is a proper noun.

We have developed a novel probabilistic model that takes into account features of the word in determining the likelihood of the word given a part of speech. This was used instead of the "word emit" probabilities $p(w_i \mid t_i)$ for known words. To estimate $p(w_i \mid t_i)$ for an unknown word, we first determined the features we thought would distinguish parts of speech. There are four independent categories of features: inflectional endings, derivational endings, hyphenation, and capitalization; these are not necessarily independent, though we are treating them as such for our tests. Our initial test had 3 inflectional endings (*-ed, -s, -ing*), and 32 derivational endings (including *-ion, -al, -ive, -ly*). Capitalization has four values, in our system (+ initial + capitalized, − initial + capitalized, etc.) in order to take into account the first word of a sentence. We can incorporate these features of the word into the probability that this particular word will occur given a particular tag using the following:

$$p(w_i \mid t_i) = p(\text{unknown-word} \mid t_i) * p(\text{Capital} - \text{feature} \mid t_i) * p(\text{endings/hyph} \mid t_i)$$

We estimate the probability of each ending for each tag directly from supervised training data. While these probabilities are not strictly independent, the approximation is good enough to make a marked difference in classification of unknown words. As the results in Figure 2 show, the use of orthographic endings of words reduces the error rate on the unknown words by a factor of three.

We tested capitalization separately, since some data, such as that in the Third Message Understanding Conference (Sundheim 1991) is uppercase only. Titles and bibliographies will cause similar distortions in a system trained on mixed case and using capitalization as a feature. Furthermore, some languages, such as Japanese, have no explicit marking of proper nouns. Interestingly, the capitalization feature contributed very little to the reduction in error rates, whereas using the word features contributed a great deal. However, it does undeniably reduce confusion with respect to the proper noun category.

Some well-known previous efforts (Church 1988; de Marcken 1990) have dealt with unknown words using various heuristics. For instance, Church's program PARTS has a prepass prior to applying the tri-tag probability model that predicts proper nouns based on capitalization. The new aspects of our work are (1) incorporating the treatment of unknown words uniformly within the probability model, (2) approximating the component probabilities for unknowns directly from the training data, and (3) measuring the contribution of the tri-tag model, of the ending, and of capitalization.

In sum, adding a probability model of typical endings of words to the tri-tag model has yielded an accuracy of 82% for unknown words. Adding a model of capitalization to the other two models further increased the accuracy to 85%. The total effect of BBN's model has been a reduction of a factor of five in the error rate of the best previously reported performance.

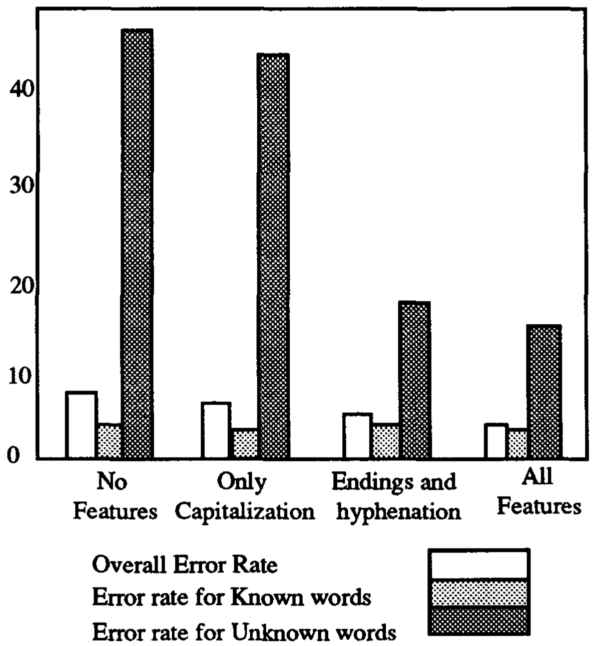


Figure 2
Decreasing error rate with use of word features.

2.4 K-best Tag Sets

An alternative mode of running POST is to return the set of most likely tags for each word, rather than a single tag for each.

In our first test, the system returned the sequence of most likely tags for the sentence. This has the advantage of eliminating ambiguity; however, even with a rather low error rate of 3.7%, there are cases in which the system returns the wrong tag, which can be fatal for a parsing system trying to deal with sentences averaging more than 20 words in length.

De Marcken (1990) developed an approximate method for finding multiple tags for each word given the preceding words and one following word. We addressed this problem by adding the ability of the tagger to return for each word an ordered list of tags, marked by their probability using the Forward Backward algorithm (Baum and Eagon 1967). That yields a more precise method of determining the probability of each possible tag since it sums over all possible tag sequences, taking into account the entire sentence and not just the preceding tags. The Forward Backward algorithm is normally used in unsupervised training to estimate the model that finds the maximum likelihood of the parameters of that model. The exact probability of a particular tag given a particular word is computed directly by the product of the "forward" and "backward" probabilities to that tag, divided by the probability of the word sequence given this model.

Figure 3 shows k-best tagging output, with the correct tag for each word marked in bold. Note that the probabilities are in natural log base e. Thus for each difference of 1, there is a factor of 2.718 in the probability.

In two of the words ("Controls" and "computerized"), the first tag is not the

Bailey Controls, based in Wickliffe Ohio, makes computerized industrial controls systems.

Bailey (NP . -1.17) (RB . -1.35) (FW . -2.32) (NN . -2.93) (NPS . -2.95) (JJS . -3.06) (JJ . -3.31) (LS . -3.41) (JJR . -3.70) (NNS . -3.73) (VBG . -3.91)...

Controls (VBZ . -0.19) (NNS . -1.93) (NPS . -3.75) (NP . -4.97)

based (VBN . -0.0001)

in (IN . -.001) (RBV . -7.07) (NP . -9.002)

Wickliffe (NP . -0.23) (NPS . -1.54)

Ohio (NP . -0.0001)

makes (VBZ . -0.0001)

computerized (VBN . -0.23) (JJ . -1.56)

industrial (JJ . -0.19) (NP . -1.73)

controls (NNS . -0.18) (VBZ . -1.77)

systems (NNS . -0.43) (NPS . -1.56) (NP . -1.95)

Figure 3
K-best tags and probabilities.

correct one. However, in all instances the correct tag is included in the set. Note the first word, "Bailey," is unknown to the system, therefore, all of the open class tags are possible.

In order to reduce the ambiguity further, we tested various ways to limit how many tags were returned based on their probabilities. Often one tag is very likely and the others, while possible, are given a low probability, as in the word "in" above. Therefore, we tried removing all tags whose probability was less than some arbitrary threshold (similar to de Marcken's "factor"), for example removing all tags whose likelihood is more than e^2 less likely than the most likely tag. So only tags within the threshold 2.0 of the most likely would be included (i.e., if the most likely tag had a log probability of -0.19, only tags with a log probability greater than -2.19 would be included). This reduced the ambiguity for known words from 1.93 tags per word to 1.23, and for unknown words, from 15.2 to 2.0.

However, the negative side of using cutoffs is that the correct tag may be excluded. Note that a threshold of 2.0 would exclude the correct tag for the word "Controls" above. By changing the threshold to 4.0, we are sure to include all the correct tags in this example, but the ambiguity for known words increases from 1.23 to 1.24, and for unknown words from 2.0 to 3.7, for an ambiguity rating of 1.57 overall.

We are continuing experiments to determine the most effective way of limiting the number of tags returned, and hence decreasing ambiguity, while ensuring that the correct tag is likely to be in the set. Balancing the tradeoff between ambiguity and accuracy is very dependent on the use the tagging will be put to. It is dependent both on the component that the tagged text directly feeds into, such as a parser that can efficiently follow many parses, but cannot recover easily from errors versus one capable of returning a partial parse, and on the application, such as an application requiring high accuracy (database query) versus one requiring high speed (processing newswire text as it comes in).

2.5 Moving to a New Domain

In all of the tests discussed so far, we both trained and tested on sets of articles in the same domain, the *Wall Street Journal* subset of the Penn TREEBANK Project. However, an important measure of the usefulness of the system is how well it performs in other domains. While we would not expect high performance in radically different kinds of text, such as transcriptions of conversations or technical manuals, we would hope for similar performance on newspaper articles from different sources and on other topics.

We tested this hypothesis using data from the Third Message Understanding Conference (MUC-3). The goal of MUC-3 was to extract data from texts on terrorism in Latin American countries. The texts are a mixture of news, interviews, and speeches. The University of Pennsylvania TREEBANK project tagged 400 MUC messages (approximately 100,000 words), which we divided into 90% training and 10% testing.

For our first test, we used the original probability tables trained from the *Wall Street Journal* articles, but tested on MUC messages. We then retrained the probabilities on the MUC messages and ran a second test on MUC messages, with an average improvement of three percentage points in both bi- and tri- tags. The full results are shown in Figure 4; 8.5% of the words in the test were unknown:

While the results using the new tables are an improvement in these first-best tests, we saw the best results using K-best mode, which obtained a .7% error rate. We ran several tests using our K-best algorithm with various thresholds. As described in Section 2.4, the threshold limits how many tags are returned based on their probabilities. While this reduces the ambiguity compared to considering all possibilities, it also increases the error rate. Figure 5 shows this tradeoff from effectively no threshold, on the right-hand side of the graph (shown in the figure as a threshold of 12), which has a .7% error rate and an ambiguity of 3, through a cutoff of 2, which has an error rate of 2.9, but an ambiguity of nearly zero—i.e., one tag per word. (Note that the far left of the graph is the error rate for a cutoff of 0, that is, only considering the first of the k-best tags, which is approximately the same as the bi-tag error rate shown in Figure 4.)

2.6 Using Dictionaries

In all of the results reported here, we are using word/part-of-speech tables derived from training, rather than on-line dictionaries to determine the possible tags for a given word. The advantage of the tables is that the training provides the probability of a word given a tag, whereas the dictionary makes no distinctions between common and uncommon uses of a word. The disadvantage of this is that uses of a word that did not occur in the training set will be unknown to the system. For example, in the training portion of the *WSJ* corpus, the word "put" only occurred as a verb. However, in our test set, it occurred as a noun in the compound "put option." Since for efficiency reasons, we only consider those tags known to be possible for a word, this will cause an error.

We have since integrated on-line dictionaries into the system, so that alternative word senses will be considered, while still not opening the set of tags considered for a known word to all open class tags. This will not completely eliminate the problem, since words are often used in novel ways, as in this example from a public radio plea for funds: "You can Mastercard your pledge."

BI-TAGS:	TEST 1	TEST 2
Overall error rate:	8.5	5.6
Number of correct tags:	10340	10667
Number of incorrect tags:	966	639
Error rate for known words:	6.3	4.6
Error rate for unknown words:	25	16
TRI-TAGS:		
Overall error rate:	8.3	5.7
Number of correct tags:	10358	10651
Number of incorrect tags:	948	655
Error rate for known words:	5.9	4.6
Error rate for unknown words:	26	18

Figure 4
Comparison of original and trained probabilities.

3. Parsing with a Context-Free Backbone

The performance of today's natural language understanding systems is hindered by the following three complementary problems:

1. Frequently more than one interpretation remains even after all linguistic and domain knowledge has been used in processing an input.
2. Partial interpretation, when no complete interpretation can be found, is minimal.
3. Finding any interpretation if the input includes an unknown word.

Our results on problems (1) and (3) above are presented in this section. The problem of partial interpretation when no complete interpretation can be found is touched upon in Section 4.

Probabilities can quantify the likelihood of alternative complete interpretations of a sentence. In these experiments, we used the grammar of the Delphi component from BBN's HARC system (Stallard 1989), which combines syntax and semantics in a unification formalism. We employed a *context-free* model, which estimates the probability

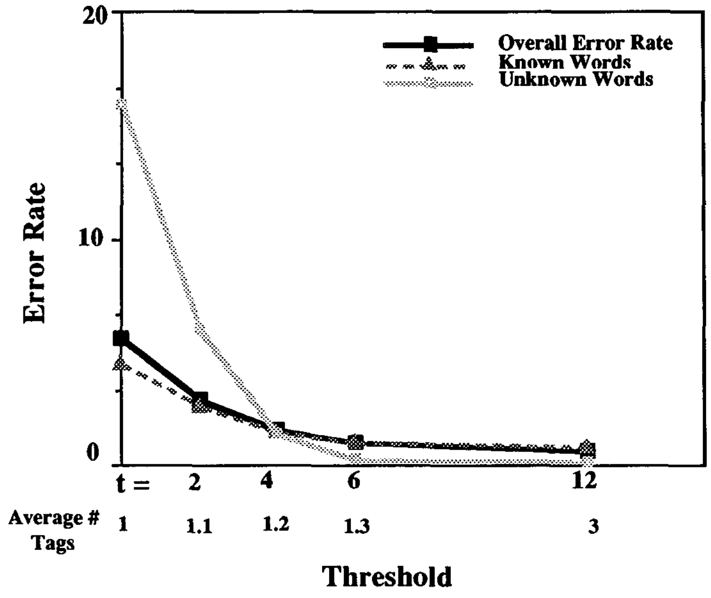


Figure 5
Comparison of thresholds for K-best.

of each rule in the grammar independently (in contrast to a context-sensitive model, such as the tri-tag model described above, which bases the probability of a tag on what other tags are in the adjacent context).

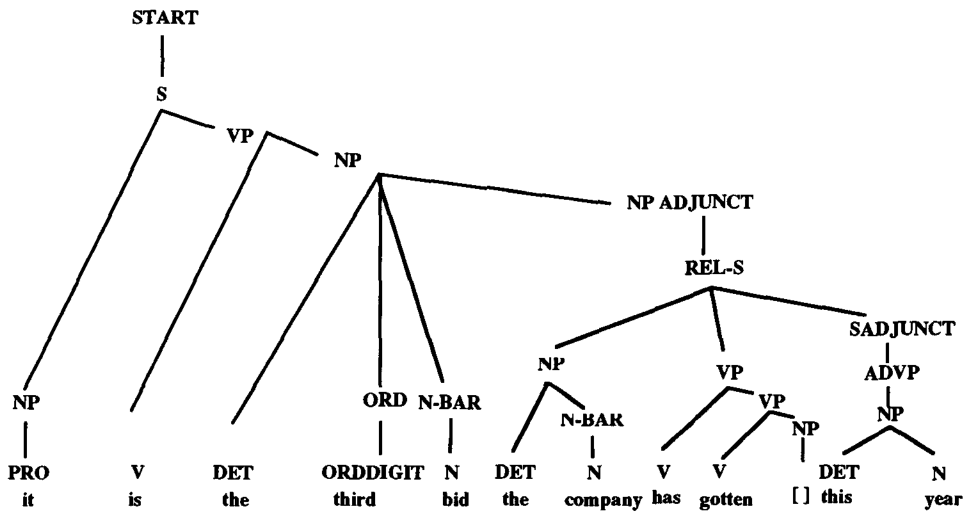
In the context-free model, we associate a probability with each rule of the grammar. For each distinct major category (left-hand side) of the grammar, there is a set of context-free rules

$$\begin{array}{lcl} \text{LHS} & \leftarrow & \text{RHS}_1 \\ \text{LHS} & \leftarrow & \text{RHS}_2 \\ & \bullet\ \bullet\ \bullet & \\ \text{LHS} & \leftarrow & \text{RHS}_n. \end{array}$$

For each rule, one estimates the probability of the right-hand side given the left-hand side, $p(\text{RHS}_j \mid \text{LHS})$. With supervised training, where a set of correct parse trees is provided as training, one estimates $p(\text{RHS}_j \mid \text{LHS})$ by the number of times rule LHS $\leftarrow$ RHS$_j$ appears in the training set divided by the number of times LHS appears in the trees.

The probability of a syntactic structure S, given the input string W, is then modeled by the product of the probabilities of the rules used in S. Chitrao and Grishman (1990) used a similar context-free model. Using this model, we explored the following issues:

- What method of training the rule probabilities should be employed? Our results were much more effective with supervised training, which explains why the model performed better in our experiments than Chitrao and Grishman found with unsupervised training.



Rules Used

START → S

S → NP VP

VP → V NP

NP → DET ORD N-BAR NPADJUNCT

⋮

$$p(TREE|W) = \prod_i p_i$$

Figure 6
Probability of a parse tree given words.

- How much (little) training data is required for reliable estimates? As little as 80 sentences of supervised training proved adequate for a grammar of about 1,000 rules.
- How is system performance influenced? Contrasted with no model, the context-free probability model reduced the error rate by a factor of two.
- Do the results suggest refinements in the probability model? Extensions to account for lowest attachment are suggested by an analysis of errors occurring in the test sets.

3.1 Selecting among Interpretations

Our intention is to use the TREEBANK corpus being developed at the University of Pennsylvania as a source of correct structures for training. However, in our first experiments, we used small training sets taken from an existing question-answering corpus of sentences about a personnel database. To our surprise, we found that as little as 80 sentences of supervised training (in which a person, using graphical tools, identifies the correct parse) are sufficient to improve the ranking of the interpretations found. In our tests, the NLP system produces all interpretations satisfying all syntactic and semantic constraints. From that set, the intended interpretation must be chosen. The context-free probability model reduced the error rate on an independent test set

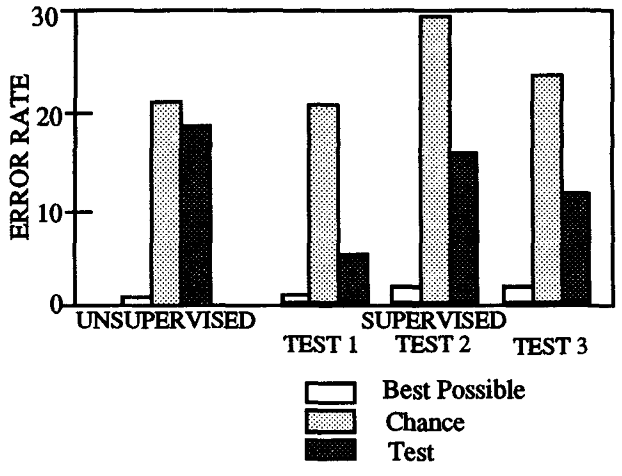


Figure 7
Predictions of probabilistic language model.

by a factor of two to four, compared with no model, i.e., random selection from the interpretations satisfying all knowledge-based constraints.

We tested the predictive power of rule probabilities using this model both in unsupervised and in supervised mode. In the former case, the input is all parse trees (whether correct or not) for the sentences in the training set. In the latter case, the training data included a specification of the correct parse as hand picked by the grammar's author from among the parse trees produced by the system.

The detailed results from using a training set of 81 sentences appear in the histogram in Figure 7. The fact that so little data was adequate deserves further scrutiny. The grammar had approximately 1,050 rules, one third of which are lexical, e.g., a category goes to a word. Estimating the lexical level is best handled via the part-of-speech techniques covered in the previous section. Therefore, there were 700 nonlexical rules. The training corpus consisted of 81 sentences whose parses averaged approximately 35 rules per sentence. Therefore, the corpus of trees included approximately 2,850 rule occurrences, or about 4 per rule on average over all rules. However, as few as half of the rules were actually employed, leading to an average of roughly 8 rule occurrences per rule observed. Therefore, there was close to the amount of data one would predict as desirable.

One further note about counting rule occurrences in the unification grammar. Rather than counting different unification bindings as different rules, we counted the rule with unbound variables, representing an equivalence class of rules with bound variables.

The "best possible" error rates for each test indicates the percentage of cases for which none of the interpretations produced by the system was judged correct, so that no selection scheme could achieve a lower error rate than that. The "chance" score gives the error rate that would be expected with random selection from all interpretations produced. The "test" column shows the error rate with the supervised or unsupervised probability model in question. The first supervised test had an 81.4% improvement, the second a 50.8% improvement, and the third a 56% improvement.

These results state how much better than chance the given model did as a percentage of the maximum possible improvement.

We expect to improve the model's performance by recording probabilities for other features in addition to just the set of rules involved in producing them. For example, in the grammar used for this test, two different attachments for a prepositional phrase produced trees with the same set of rules, but differing in shape. Thus the simple, context-free model based on the product of rule probabilities could not capture preferences concerning such attachment. By adding to the model probabilities for such additional features, we expect that the power of the probabilistic model to automatically select the correct parse can be substantially increased.

Second, a much more reliable estimate of $p(\text{word} \mid \text{category})$ can be estimated as described in Section 2. In fact, one should be able to improve the estimate of a tree's likelihood via $p(S \mid W) = p(S \mid T) * p(T \mid W)$.

3.2 Learning Lexical Syntax and Lexical Semantics

One purpose for probabilistic models is to contribute to handling new words or partially understood sentences. We have done preliminary experiments that show that there is promise in learning lexical, syntactic, and semantic features from context when probabilistic tools are used to help control the ambiguity.

In our experiments, we used a corpus of sentences each with one word that the system did not know. To create the corpus, we began with a corpus of sentences known to parse from a personnel question-answering domain. We then replaced one word in each sentence with an undefined word.

For example, in the following sentence, the word "contact" is undefined in the system: *Who in Division Four is the contact for MIT?* That word has both a noun and a verb part of speech; however, the pattern of parts of speech of the words surrounding "contact" causes the tri-tag model to return a high probability that the word is a noun. Using unification variables for all possible features of a noun, the parser produces multiple parses. Applying the context-free rule probabilities to select the most probable of the resulting parses allows the system to conclude both syntactic and semantic facts about "contact." Syntactically, the system discovers that it is a count noun, with third person singular agreement. Semantically, the system learns (from the use of "who") that "contact" is in the semantic class PERSONS.

Furthermore, the partially specified semantic representation for the sentence as a whole also shows the semantic relation to SCHOOLS, which is expressed here by the *for* phrase. Thus, even a single use of an unknown word in context can supply useful data about its syntactic and semantic features.

Probabilistic modeling plays a key role in this process. While context-sensitive techniques for inferring lexical features can contribute a great deal, they can still leave substantial ambiguity. As a simple example, suppose the word "list" is undefined in the sentence *List the employees.* The tri-tag model predicts both a noun and a verb part of speech in that position. Using an underspecified noun sense combined with the usual definitions for the rest of the words yields no parses. However, an underspecified verb sense yields three parses, differing in the subcategorization frame of the verb "list." For more complex sentences, even with this very limited protocol, the number of parses for the appropriate word sense can reach into the hundreds.

Using the rule probabilities acquired through supervised training (described in the previous section), the likelihood of the ambiguous interpretations resulting from a sentence with an unknown word was computed. Then we tested whether the tree ranked most highly matched the tree previously selected by a person as the correct one. This tree equivalence test was based on the tree's structure and on the rule applied at each node; while an underspecified tree might have some less-specified feature values than the chosen fully specified tree, it would still be equivalent in the sense above.

Of 160 inputs with an unknown word, in 130 cases the most likely tree matched the correct one, for an error rate of 18.75%, while picking at random would have resulted in an error rate of 37%, for an improvement by a factor of 2. This suggests that probabilistic modeling can be a powerful tool for controlling the high degree of ambiguity in efforts to automatically acquire lexical data.

We have also begun to explore heuristics for combining lexical data for a single word acquired from a number of partial parses. There are some cases in which the best approach is to unify the two learned sets of lexical features, so that the derived sense becomes the sum of the information learned from the two examples. For instance, the verb subcategorization information learned from one example could be thus combined with agreement information learned from another. On the other hand, there are many cases, including alternative subcategorization frames, where each of the encountered options needs to be included as a separate alternative.

4. Partial Parsing and Learning Case Frame Information

Traditionally, natural language processing (NLP) has focused on obtaining complete syntactic analyses of all input and on semantic analysis based on handcrafted knowledge. However, grammars are incomplete, text often contains new words, and there are errors in text. Furthermore, as research activities tackle broader domains, if the research results are to scale up to realistic applications, handcrafting knowledge must give way to automatic knowledge base construction.

An alternative to traditional parsers is represented in FIDDITCH (Hindle 1983), MITFP (de Marcken 1990), and CASS (Abney 1990). Instead of requiring complete parses, a forest is frequently produced, each tree in the forest representing a nonoverlapping fragment of the input. However, algorithms for finding the semantics of the whole from the disjoint fragments have not previously been developed or evaluated.

We have been comparing several differing algorithms from various sites to evaluate both the effectiveness of such a strategy in correctly predicting fragments. This is reported first (Section 5.1).

The central experiment in this section tests the feasibility of learning case frame information for verbs from examples. In the method tested, we assume that a body of fully parsed sentences, such as those from TREEBANK, are available. We furthermore assume that every head noun and head verb has a lexical link to a unary predicate in a taxonomic domain model; that unary predicate is the most specific semantic class of entities denoted by the headword. From the parsed examples and the lexical links to the domain model, an algorithm identifies case frame relations for the verbs.

4.1 Finding Core Noun Phrases

If an algorithm is to learn case frame relations (or selection restrictions) from text, a basic concern is to reliably identify noun phrases and their semantic category, even if neither full syntactic nor full semantic analysis is possible. First, we discuss reliably finding them based on local syntactic information. In the next section, we describe finding their semantic category.

Two of our experiments have focused on the identification of core noun phrases, a primary way of expressing entities in text. A core NP is defined syntactically as the maximal simple noun phrase, i.e., the largest one containing no post-modifiers.

Here are some examples of core NPs (marked by italics) within their full noun phrases:

> *a joint venture* with the Chinese government to build an automobile-parts assembly plant
>
> *a $50.9 million loss* from discontinued operations in the third quarter because of the proposed sale

Such complex, full NPs require too many linguistic decisions to be directly processed without detailed syntactic and semantic knowledge about each word, an assumption that need not be true for open-ended text.

We tested two differing algorithms on text from the *Wall Street Journal* (*WSJ*). Using BBN's part-of-speech tagger (POST), tagged text was parsed using the full unification grammar of Delphi to find only core NPs, 695 in 100 sentences. Hand-scoring of the results indicated that 85% of the core NPs were identified correctly. Subsequent analysis suggested that half the errors could be removed with only a little additional work, suggesting that over 90% performance is achievable.

In a related test, we explored the bracketings produced by Church's PARTS program (Church 1988). We extracted 200 sentences of *WSJ* text by taking every tenth sentence from a collection of manually corrected parse trees (data from the TREEBANK Project at the University of Pennsylvania). We evaluated the NP bracketings in these 200 sentences by hand and tried to classify the errors. Of 1226 phrases in the 200 sentences, 131 were errors, for a 10.7% error rate. The errors were classified by hand as follows:

- Two consecutive but unrelated phrases grouped as one: 10
- Phrase consisted of a single word, which was not an NP: 70
- Missed phrases (those that should have been bracketed but were not): 12
- Ellided head (e.g. part of a conjoined premodifier to an NP): 4
- Missed premodifiers: 4
- Head of phrase was verb form that was missed: 4
- Other: 27.

The 90% success rate in both tests suggests that identification of core NPs can be achieved using only local information and with minimal knowledge of the words. Next we consider the issue of what semantics should be assigned and how reliably that can be accomplished.

4.2 Semantics of Core Noun Phrases

In trying to extract pre-specified data from open-ended text such as a newswire, it is clear that full semantic interpretation of such texts is not on the horizon. However, our hypothesis is that it need not be for automatic data base update. The type of information to be extracted permits some partial understanding. For semantic processing, minimally, for each noun phrase (NP), one would like to identify the class in the domain model that is the smallest pre-defined class containing the NP's denotation. Since we have assumed that the lexicon has a pointer to the most specific class in the domain model, the issue reduces to whether we can algorithmically predict the word, if any, in a noun phrase that denotes the NP's semantic class. For each clause, one would like to identify the corresponding event class or state of affairs denoted.

Our pilot experiment focused on the reliability of identifying the minimal class for each noun phrase.

Assigning a semantic class to a core noun phrase can be handled via some structural rules. Usually the semantic class of the headword is correct for the semantic class not only of the core noun phrase but also of the complete noun phrase it is part of. Additional rules cover exceptions, such as "set of ...". These heuristics correctly predicted the semantic class of the whole noun phrase 99% of the time in the sample of over 1000 noun phrases from the *WSJ* that were correctly predicted by Church's PARTS program.

Furthermore, even some of the NPs whose left boundary was not predicted correctly by PARTS nevertheless were assigned the correct semantic class. One consequence of this is that the correct semantic class of a complex noun phrase can be predicted even if some of the words in the noun phrase are unknown and even if its full structure is unknown. Thus, fully correct identification of core noun phrase boundaries and of noun phrase boundaries may not be necessary to accurately produce database updates.

This result is crucial to our method of inferring case frames of verbs from examples. Simple rules can predict which word designates the semantic clause of a noun phrase very reliably. We can use these simple rules plus lexical lookup to identify the basic semantic class of a noun phrase.

4.3 Learning Semantic Information

Semantic knowledge called *selection restrictions* or *case frames* governs what phrases make sense with a particular verb or noun (what arguments go with a particular verb or noun). Traditionally such semantic knowledge is handcrafted, though some software aids exist to enable greater productivity (Ayuso, Shaked, and Weischedel 1987; Bates 1989; Grishman, Hirschman, and Nhan 1986; Weischedel et al. 1989).

Instead of handcrafting this semantic knowledge, our goal is to learn that knowledge from examples, using a three-step process:

1. Simple manual semantic annotation
2. Supervised training based on parsed sentences
3. Estimation of probabilities.

4.3.1 Simple Manual Semantic Annotation. Given a sample of text, we annotate each noun, verb, and proper noun in the sample with the semantic class corresponding to it in the domain model. For instance, *dawn* would be annotated <time>, *explode* would be <explosion event>, and *Yunguyo* would be <city>. For our experiment, 560 nouns and 170 verbs were defined in this way. We estimate that this semantic annotation proceeded at about 90 words per hour.

4.3.2 Supervised Training. From the TREEBANK project at the University of Pennsylvania, we used 20,000 words of MUC-3 texts that had been bracketed according to major syntactic category. The bracketed constituents for the sentence below appears in Figure 8.

A bomb exploded today at dawn in the Peruvian town of Yunguyo, near the lake, very near where the Presidential summit was to take place.

```
((S
   (NP a bomb)
   (VP exploded
       today
       (PP at
          (NP dawn) )
       (PP in
           (NP the Peruvian town
               (PP of
                   (NP yunguyo) ) ) )
      ,
      (PP near
              (NP the lake) )
      ,
      (SBAR ( WHPP very
                     near
                      (WHADVP where) )
              (S (NP the presidential summit)
                  (VP was
                 (S (NP*)
                     to
                     (VP take
                          (NP place) ) ) ) ) ) ) )
 .)
```

Figure 8
Example of TREEBANK analysis.

From the example one can clearly infer that bombs can explode, or more properly, that *bomb* can be the logical subject of *explode,* that *at dawn* can modify explode, etc. Naturally good generalizations based on the instances are more valuable than the instances themselves.

Since we have a hierarchical domain model, and since the manual semantic annotation states the relationship between lexical items and concepts in the domain model, we can use the domain model hierarchy as a given set of categories for generalization. However, the critical issue is selecting the right level of generalization given the set of examples in the supervised training set.

We have chosen a known statistical procedure (Katz 1987) that selects the minimum level of generalization such that there is sufficient data in the training set to support discrimination of cases of attaching phrases (arguments) to their head. This leads us to the next topic, estimation of probabilities from the supervised training set.

4.3.3 Estimation of Probabilities. The case relation, or selection restriction, to be learned is of the form X P O, where X is a headword or its semantic class; P is a case, e.g., logical subject, logical object, preposition, etc.; and O is a head word or its semantic class.

One factor in the probability that O attaches to X with case P is $p'(X \mid P, O)$, an estimate of the likelihood of attaching PO to X given P and O. We chose to model a second multiplicative factor $p(d)$, the probability of an attachment where d words separate the headword X from the phrase to be attached (intuitively, the notion of attachment distance). For instance, in the example previously discussed, *in the town* is attached to the verb *explode*, at a distance of four words; *of Yunguyo* is attached to the noun *town* at a distance of one word back, etc. Thus we estimate the probability of attachment as $p'(X \mid P, O) * p(d)$.

Since a 20,000-word corpus does not constitute enough data to estimate the probability of all triples, we used an extension and generalization of an algorithm (Katz 1987) to automatically move up the hierarchical domain model from X to its parent, and from O to its parent. The "backing-off" that was originally proposed for the estimation of probabilities of n-gram sequences of words starts with the most detailed model. In this case we start with the explicit probability of the phrase PO attaching to the word X. If we have no examples of X P O in the training set we consider with some penalty a class of X or O. Thus the event becomes less specific but more likely to have been observed. We back off on the detail until we can estimate the probability from the training set. The Katz algorithm gives a way to estimate the back-off penalty as the probability that we would not have observed the more detailed triple even though it was possible.

4.3.4 The Experiment. By examining the table of triples X P O that were learned, it was clear that meaningful information was induced from the examples. For instance, (<attack> *against* <building>) and (<attack> *against* <residence>) were learned, which correspond to two cases of importance in the MUC domain. As a consequence, useful semantic information was learned by the training algorithm.

However, we ran a far more meaningful evaluation of what was learned by measuring how effective the learned information would be at predicting 166 prepositional phrase attachments that were not made by our partial parser. For example, in the following sentence, *in the Peruvian town* can be attached syntactically at three places: modifying *dawn*, modifying *today*, or modifying *explode*.

A bomb exploded today at dawn in the Peruvian town of Yunguyo, near the lake, very near where the Presidential summit was to take place.

Closest attachment, a purely syntactic constraint, worked quite effectively, having a 25% error rate. Using the semantic probabilities alone $p'(X \mid P, O)$ had poorer performance, a 34% error rate. However, the richer probability model $p'(X \mid P, O) * p(d)$ outperformed both the purely semantic model and the purely syntactic model (closest attachment), yielding an 18% error rate.

However, the degree of reduction of error rate should not be taken as the final word, for the following reasons:

- 20,000 words of training data is much less than one would want.
- Since many of the headwords in the 20,000 word corpus are not of import in the MUC-3 domain, their semantic type is vague, i.e., <unknown event>, <unknown entity>, etc.

4.4 Related Work

In addition to the work discussed earlier on tools to increase the portability of natural language systems, another recent paper (Hindle and Rooth 1990) is directly related to our goal of inferring case frame information from examples.

Hindle and Rooth focused only on prepositional phrase attachment using a probabilistic model, whereas our work applies to all case relations. Their work used an unsupervised training corpus of 13 million words to judge the strength of prepositional affinity to verbs, e.g., how likely it is for *to* to attach to the word *go,* for *from* to attach to the word *leave,* or for *to* to attach to the word *flight*. This lexical affinity is measured independently of the object of the preposition. By contrast, we are exploring induction of semantic relations from supervised training, where very little training may be available. Furthermore, we are looking at triples of headword (or semantic class), syntactic case, and headword (or semantic class).

In Hindle and Rooth's test, they evaluated their probability model in the limited case of verb–noun phrase–prepositional phrase. Therefore, no model at all would be at least 50% accurate. In our test, many of the test cases involved three or more possible attachment points for the prepositional phrase, which provided a more realistic test.

An interesting next step would be to combine these two probabilistic models (perhaps via linear weights) in order to get the benefit of domain-specific knowledge, as we have explored, and the benefits of domain-independent knowledge, as Hindle and Rooth have explored.

4.5 Future Work: Finding Relations/Combining Fragments

The experiments on the effectiveness of finding core NPs using only local information were run by midsummer 1990. In fall 1990, another alternative, the Fast Partial Parser (FPP), which is a derivative of earlier work (de Marcken 1990), became available to us. It finds fragments using a stochastic part of speech algorithm and a nearly deterministic parser. It produces fragments averaging three to four words in length. Figure 9 shows an example output for the sentence.

> *A BOMB EXPLODED TODAY AT DAWN IN THE PERUVIAN TOWN OF YUNGUYO, NEAR THE LAKE, VERY NEAR WHERE THE PRESIDENTIAL SUMMIT WAS TO TAKE PLACE.*

Certain sequences of fragments appear frequently, as illustrated in Tables 1 and 2. One frequently occurring pair is an S followed by a PP (prepositional phrase). Since there is more than one way the parser could attach the PP, and syntactic grounds alone for attaching the PP would yield poor performance, semantic preferences applied by a post-process that combines fragments are called for.

In our approach, we propose using local syntactic and semantic information rather than assuming a global syntactic and semantic form will be found. The first step is to compute a semantic interpretation for each fragment found without assuming that the meaning of each word is known. For instance, as described above, the semantic class for any noun phrase can be computed provided the head noun has semantics in the domain.

Based on the data above, a reasonable approach is an algorithm that moves left-to-right through the set of fragments produced by FPP, deciding to attach fragments (or not) based on semantic criteria. To avoid requiring a complete, global analysis, a window two constituents wide is used to find patterns of possible relations among phrases. For example, an S followed by a PP invokes an action of finding all points along the "right edge" of the S tree where a PP could attach, applying the fragment combining patterns at each such spot, and ranking the alternatives.

As is evident in Table 2, FPP frequently does not attach punctuation. This is to be expected, since punctuation is used in many ways, and there is no deterministic basis for attaching the constituent following the punctuation to the constituent preceding it.

```
(S (NP (DETERMINER "A") (N "BOMB"))
   (VP (AUX (NP (MONTH "TODAY"))
            (PP (PREP "AT")
                (NP (N "DAWN"))))
       (VP (V "EXPLODED"))))
(PP
  (PP (PREP "IN")
      (NP (NP (DETERMINER "THE")
              (N "PERUVIAN")
              (N "TOWN"))
          (PP (PREP "OF")
              (NP (N "YUNGUYO")))))
  (PUNCT ","))
(PP (PP (PREP "NEAR")
        (NP (DETERMINER "THE")
            (N "LAKE")))
    (PUNCT ","))
(ADJP (DEGREESPEC "VERY")
      (ADJP (ADJ "NEAR")))
(ADV "WHERE")
(NP (DETERMINER "THE")
    (ADJP (ADJ "PRESIDENTIAL"))
    (N "SUMMIT"))
(VP (AUX) (VP (V "WAS")))
(VP (AUX "TO")
    (VP (V "TAKE")
        (NP (N "PLACE"))))
(PUNCT ".")
```

Figure 9
Example output.

Table 1
Most frequently occurring pairs (in 2500 pairs).

Pair		Occurrences
S	PP	104
NP	VP	89
VP	VP	72
S	VP	65
PP	PP	62
PP	NP	58
NP	PP	56
VP	PP	54
PP	VP	48
NP	NP	34

Therefore, if the pair being examined by the combining algorithms ends in punctuation, the algorithm looks at the constituent following it, trying to combine it with the constituent to the left of the punctuation.

A similar case is when the pair ends in a conjunction. Here the algorithm tries to combine the constituent to the right of the conjunction with that on the left of the conjunction.

Table 2
Frequently occurring fragment pairs surrounding punctuation.

Triple			Occurrences
NP	PUNCT	NP	53
VP	PUNCT	S	20
S	PUNCT	S	19
NP	PUNCT	S	19
S	PUNCT	NP	17
VP	PUNCT	N	12
NP	PUNCT	PP	10
NP	PUNCT	VP	9

5. Conclusions

Our pilot experiments indicate that a hybrid approach to text processing including corpus-based probabilistic models to supplement knowledge-based techniques is both feasible and promising.

In part-of-speech labeling, we have evaluated POST in the laboratory, evaluating its results against the work of people doing the same task. However, the real test of such a system is how well it functions as a component in a larger system. Can it make a parser work faster and more accurately? Can it help to extract certain kinds of phrases from unrestricted text? We are currently running these experiments by making POST a part of existing systems. It was run as a preprocessor to Grishman's Proteus system for the MUC-3 competition (Grishman and Sterling 1989). Preliminary results showed it sped up Proteus by a factor of two in one-best mode and by a factor of 33% with a threshold of T=2. It is integrated into a new message processing system (PLUM) at BBN.

For reducing interpretation ambiguity, our context-free probability model, trained in supervised mode on only 81 sentences, was able to reduce the error rate for selecting the correct parse on independent test sets by a factor of 2–4. For the problem of processing new words in text, the probabilistic model reduced the error rate for picking the correct part of speech for such words from 91.5% to 15%. And once the possible parts of speech for a word are known (or hypothesized using the tri-tag model), the probabilistic language model proved useful in indicating which parses should be looked at for learning more complex lexical information about an unknown word. However, the most innovative aspect of our approach is the automatic induction of semantic knowledge from annotated examples. The use of probabilistic models offers the induction procedure a decision criterion for making generalizations from the corpus of examples.

Acknowledgments
The work reported here was supported by the Defense Advanced Research Projects Agency and was monitored by Rome Laboratory under Contract No. F30602-87-D-0093 and Contract No. F30602-91-C-0051. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, whether expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

References

Abney, S. (1990). "Rapid incremental parsing with repair." In *Proceedings, Sixth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research.* 1–9.

Ayuso, D. M.; Shaked, V.; and Weischedel, R. M. (1987). "An environment for acquiring semantic information." In *Proceedings, 25th Annual Meeting of the Association for Computational Linguistics.* 32–40.

Bates, M. (1989). "Rapid porting of the Parlance^{TM} natural language interface." In *Proceedings, Speech and Natural Language Workshop.* Morgan Kaufmann Publishers. 83–88.

Baum, L. E., and Eagon, J. A. (1967). "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model of ecology." *Amer. Math Soc. Bulletin,* 73, 360–362.

Chitrao, M. V., and Grishman, R. (1990). "Statistical parsing of messages." In *Proceedings, Speech and Natural Language Workshop.* Morgan Kaufmann Publishers. 263–266.

Church, K. (1988). "A stochastic parts program and noun phrase parser for unrestricted text." In *Proceedings, Second Conference on Applied Natural Language Processing.* 136–143.

de Marcken, C. G. (1990). "Parsing the LOB corpus." In *Proceedings, 28th Annual Meeting of the Association for Computational Linguistics.* 243–251.

DeRose, S. J. (1988). "Grammatical category disambiguation by statistical optimization." *Computational Linguistics,* 14, 31–39.

Grishman, R., and Sterling, J. (1989). "Preference semantics for message understanding." In *Proceedings, Speech and Natural Language Workshop.* Morgan Kaufmann Publishers. 71–74.

Grishman, R.; Hirschman, L.; and Nhan, N. T. (1986). "Discovery procedures for sublanguage selectional patterns: Initial experiments." *Computational Linguistics,* 12, 205–215.

Hindle, D. (1983). "User manual for Fidditch." Naval Research Laboratory Technical Memorandum #7590-142.

Hindle, D., and Rooth, M. (1990). "Structural ambiguity and lexical relations." In *Proceedings, Speech and Natural Language Workshop.* Morgan Kaufman Publishers. 257–262.

Jelinek, F. (1985). "Self organizing language modeling for speech recognition." Unpublished Technical Report, IBM T. J. Watson Research Center, Yorktown Heights, NY.

Katz, S. M. (1987). "Estimation of probabilities from sparse data for the language model component of a speech recognizer." In *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol. ASSP-35 No. 3.

Kuhn, R., and De Mori, R. (1990). "A cache-based natural language model for speech recognition." In *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12, 570–583.

Kupiec, J. (1989). "Augmenting a hidden Markov model for phrase-dependent word tagging." In *Proceedings, Speech and Natural Language Workshop.* Morgan Kaufmann Publishers. 92–98.

Marcus, M.; Santorini, B.; and Magerman (1990). "First steps towards an annotated database of American English." In *Readings for Tagging Linguistic Information in a Text Corpus,* edited by Langendoen and Marcus, Tutorial for the 28th Annual Meeting of the Association for Computational Linguistics.

Santorini, B. (1990). "Annotation manual for the Penn TREEBANK project." Technical Report, CIS Department, University of Pennsylvania.

Stallard, D. (1989). "Unification-based semantic interpretation in the BBN spoken language system." In *Proceedings, Speech and Natural Language Workshop.* Morgan Kaufmann Publishers. 39–46.

Sundheim, B. M. (1991). "Overview of the Third Message Understanding Evaluation and Conference." In *Proceedings, Third Message Understanding Conference (MUC-3).* Morgan Kaufmann Publishers, 3–16.

Weischedel, R. M.; Bobrow, R.; Ayuso, D. M.; and Ramshaw, L. (1989). "Portability in the Janus natural language interface." In *Speech and Natural Language.* Morgan Kaufmann Publishers. 112–117.