POST: Using Probabilities in Language Processing

Marie Meteer, Richard Schwartz, Ralph Weischedel

BBN Systems and Technologies
10Moulton Street
Cambridge, MA 02138
U.S.A.

Abstract

We report here on our experiments with POST (Part of Speech Tagger) to address problems of ambiguity and of understanding unknown words. Part of speech tagging, per se, is a well understood problem. Our paper reports experiments in three important areas: handling unknown words, limiting the size of the training set, and returning a set of the most likely tags for each word rather than a single tag. We describe the algorithms that we used and the specific results of our experiments on Wall Street Journal articles and on MUC terrorist messages.

1. Introduction[1]

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[2], Parlance™[3] and Delphi[4], use these techniques quite successfully. However, as we move from the problem of understanding queries in fixed domains to processing open text for applications such as data extraction, we have found rule-based techniques too brittle, and the amount of work necessary to build them intractable, especially when attempting to use the same system on multiple domains.

We report in this paper on one application of probabilistic models to language processing, the assignment of part of speech to words in open 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 two ways: (1) We have run experiments regarding the amount of training data needed in moving to a new domain; (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. We describe POST and its algorithms and then we describe our extensions, showing the results of our experiments.

[1] The work reported here was supported by the Advanced Research Projects Agency and was monitored by the Rome Air Development Center under Contract No. F30602-87-D-OO93. 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.

[2] Weischedel, et al. 1989.

[3] Parlance is a trademark of BBN Systems and Technologies.

[4] Stallard, 1989.

2. POST: Using probabilities to tag part of speech

Predicting the part of speech of a word is one straightforward way to use probabilities. Many words are several ways ambiguous, such as the following:

around 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." where both words are two ways ambiguous, resulting in two grammatical interpretations as sentences.

Models predicting part of speech can serve to cut down the search space a parser must consider in processing known words and make the selection among alternatives more accurate. Furthermore, they can be used as one input to more complex strategies for inferring lexical and semantic information about unknown words.

2.1 The n-gram model

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, given a particular word sequence, W. Using Bayes' rule we can write the *a posteriori* probability of tag sequence T given word sequence as

$$P(W|T) = \frac{P(T)P(W)}{P(W)}$$

where P(T) is the a priori probability of tag sequence T, P(WIT) 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 the a posteriori probability of each, and choose the one that is highest. 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.

$$P(T|W)\,P(W) = \prod p(t_0)\; p(t_1 \mid t_0)\; p(t_i \mid t_{i-1}, t_{i-2}, \ldots)\; p(w_i \mid t_i .., w_{i-1} ..)$$

Now, we can make the approximation that each tag depends only the immediately preceding tags (say the two preceding tags for a tri-tag model), and that the word depends only on the tag.

$$P(T|W)\,P(W) = p(t_0)\; p(t_1 \mid t_0) \prod p(t_i \mid t_{i-1}, t_{i-2})\; p(w_i \mid t_i)$$

That is, once we know the tag that will be used, we gain no further information about the likely word from knowing the previous tags or words. This model is called a Markov model, and the assumption is frequently called the Markov independence assumption.

If we have sufficient training data then we can estimate the tag n-gram sequence probabilities and the probability of each word given a tag (lexical probabilities). We use robust estimation techniques that take care of the cases of unobserved events (i.e. sequences of tags that have not occurred in the training data). However, in real-world problems, we also are likely to have words that were never observed at all in the training data. The model given above can still be used, simply by defining a generic new word called "unknown-word". The system can then guess at the tag of the unknown word primarily using the tag sequence probabilities. We return to the problem of unknown words in Section 3.

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. The UPenn corpus, which was created as part of the TREEBANK project [Santorini, 1990] consists of *Wall Street Journal* (WSJ) articles. Each word or punctuation mark has been tagged with one of 47 parts of speech[5], as shown in the following example:

Terms/NNS were/VBD not/RB disclosed/VBN ./.[6]

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, and then derives from those counts a probabilistic tri-tag model. 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. The probabilities were padded to avoid setting the probability for unseen tri-tags or unseen word senses to zero.

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 the words in the corpus are known. This is in fact about the rate of discrepancies among human taggers on the TREEBANK project [Marcus *et al.*, 1990).

2.2 Quantity of training data

While supervised training is shown here to be very effective, it requires a correctly tagged corpus. The perceived size of the corpus required affected de Marcken's choice of a bi-tag rather than a tri-tag model [de Marcken, 1990] We have done some experiments to quantify how much tagged data is really necessary.

In these 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 10 instances of each type of tag sequence in order for their probabilities to be estimated with reasonable accuracy. This would imply that a tri-tag model using 47 possible parts of speech would need a bit more than a million words of training. However, we found that much less training data was necessary.

It can be shown that if the average number of tokens of each tri-gram that has been observed is 10, 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-lag sequences actually occur. Out of about 97,000 possible triples, we found only 6170 unique triples when we trained on 64,000 words, and about 10,000 when we trained on 1,000,000 words. Thus, even 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. All that is really necessary, recalling the rule of thumb, is enough

[5] Of the 47 parts of speech, 36 are word tags and 11 punctuation tags. Of the word tags, 22 are tags for open class words and 14 for closed class words.

[6] NNS is plural noun; VBD is past tense verb; RB is adverbial; VBN is past participle verb.

training to allow for 10 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 10 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.[7]

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 2nd Message Understanding Conference held at the Naval Ocean Systems Center, June, 1989, would simply halt processing a sentence when a new word was encountered.

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 slating an action or stale 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.

Using the UPenn set of parts of speech, unknown words can be in any of the 22 open-class parts of speech. The 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 rale was 75% [Kuhn & de Mori, 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, the 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.

' 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 $p(wj \mid tj)$ 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 $p(wj \mid tj)$. 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 employed a 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[8] categories of features: inflectional endings, derivational endings, hyphenation, and capitalization. Our initial test had three 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

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

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 1 shows, the use of the orthographic endings of the words reduces the error rate on the unknown words by a factor of 3.

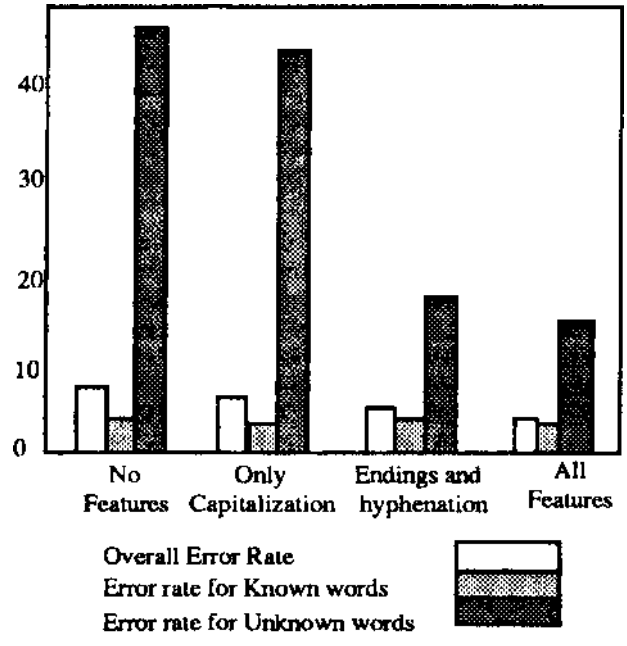

Figure 1: Decreasing error rate with use of word features

We tested capitalization separately, since some data, such as that in the Third Message Understanding Conference, is upper case only. Titles and bibliographies will cause similar distortions in a system trained on mixed case and

[8] These are not necessarily independent, though we are treating them as such for our tests.

using capitalization as a feature. Interestingly, the capitalization feature contributed very little to the reduction in error rates, whereas using the word features contributed a great deal.

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.

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 & 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.

The following example 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.

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 2: K-best Tags and Probabilities

In two of the words ("Controls" and "computerized") the first tag is not the 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 cut offs is that the correct tag may be excluded. Note that a cut off of 2.0 would exclude the correct tag for the word "Controls" above. By changing the cut off to 4.0, we are sure to include all the correct tags in this example, but the ambiguity for known words raises 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, 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 vs. one capable of returning a partial parse, and on the application, such as an application requiring high accuracy (data base query) vs. one requiring high speed (processing newswire text as it comes in).

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 texts used in 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 is to extract data from texts on terrorism in Latin American countries. The texts are mainly newspaper articles, although there are some transcriptions of interviews and speeches. The University of Pennsylvania TREEBANK project tagged four hundred 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 on the Wall Street Journal articles. We then retrained the probabilities on the MUC messages and ran a second test, with an average improvement of three percentage points in both bi- and tri- tags. The full results are shown below; 8.5% of the words in the test were unknown:

BITAGS:	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
TRTTAGS:		
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 3: Comparison of original and trained probabilities

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 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 4 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 cut off of 2, which has a error rate of 2.9, but an ambiguity of nearly zero-i.e. one tag pre word. (Note the far left of the graph is the error rate for a cut off 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 3.)

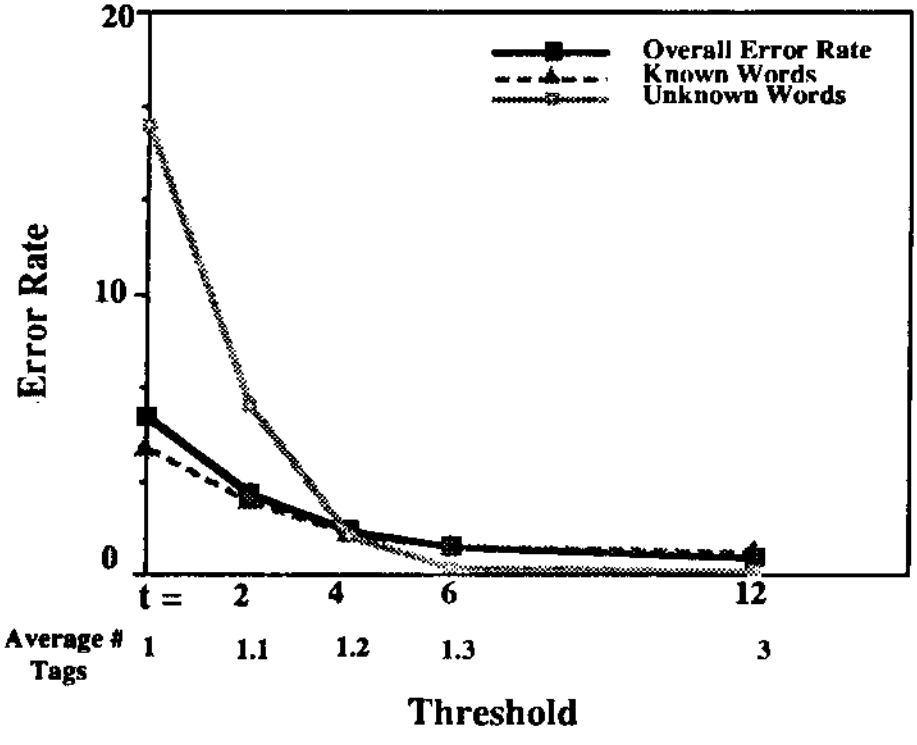

Figure 4: Comparison of thresholds for K- Best

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 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 are currently integrating 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.". We will be rerunning the experiments reported here to evaluate the effect of using on-line dictionaries.

7. Future Directions

In the work reported here, 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 is being run as a preprocessor to Grishman's Proteus system for the MUC-3 competition [Grishman & 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 also being integrated into a new message processing system at BBN. The results of these experiments will provide us with new directions and ideas both for improving POST and for other ways to integrate probabilistic models into natural language processing systems.

Acknowledgements

We would like to acknowledge Lance Ramshaw for his work on POST and the results on the size of the training set. We have also benefited from discussions with Ken Church.

References

Baum, L.E. and Eagon, J.A. 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, 1967.

Church, K. A. Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. *Proceedings of the Second Conference on Applied Natural Language Processing,* pages 136-143. ACL, 1988.

de Marcken, C.G. Parsing the LOB Corpus. *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics,* pages 243-251. 1990.

DeRose, S.J. Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics* 14: 31-39, 1988.

Grishman, R., and J. Sterling. Preference Semantics for Message Understanding. *Proceedings of the Speech and Natural Language Workshop,* pages 71-74, Oct 1989.

Jelinek, F. Self Organizing Language Modeling for Speech Recognition. Unpublished Technical Report, 1985. IBM TJ. Watson Research Center, Yorktown Heights, N.Y.

Kuhn, R., and De Mori, R., A cache-Based Natural Language Model for Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12, pages 570-583. 1990.

Kupiec, J. Augmenting a Hidden Markov Model for Phrase-Dependent Word Tagging. In *Proceedings of the Speech and Natural Language Workshop,* pages 92-98. Oct. 1989.

Santorini, B. *Annotation Manual for the Penn Treebank Project.* Technical Report. CIS Department. University of Pennsylvania. May 1990.

Stallard, D. Unification-Based Semantic Interpretation in the BBN Spoken Language System. In *Proceedings of the Speech and Natural Language Workshop,* pages 39-46. Oct. 1989.

Marcus, M., Santorini, B. & Magerman, "First Steps Towards an Annotated Database of American English" in Langendoen & Marcus, "Readings for Tagging Linguistic Information in a Text Corpus", tutorial for the 28th Annual Meeting of the Association for Computational Linguistics. June 1990.

Weischcdel, R., Bobrow, R., Ayuso, D., and Ramshaw, L. Portability in the Janus Natural Language Interface, in *Speech and Natural Language,* Morgan Kaufman Publishers, Inc. p. 112-117. Oct. 1989.