S. M. Thede and M. P. Harper, "Second-Order Hidden Markov Model for Part-of-Speech Tagging," Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics, Baltimore MD, June 1999, pp. 175-182.

A Second-Order Hidden Markov Model for Part-of-Speech Tagging

Scott M. Thede and **Mary P. Harper**
School of Electrical and Computer Engineering, Purdue University
West Lafayette, IN 47907
{*thede, harper*}*@ecn.purdue.edu*

Abstract

This paper describes an extension to the hidden Markov model for part-of-speech tagging using second-order approximations for both contextual and lexical probabilities. This model increases the accuracy of the tagger to state of the art levels. These approximations make use of more contextual information than standard statistical systems. New methods of smoothing the estimated probabilities are also introduced to address the sparse data problem.

1 Introduction

Part-of-speech tagging is the act of assigning each word in a sentence a *tag* that describes how that word is used in the sentence. Typically, these tags indicate syntactic categories, such as noun or verb, and occasionally include additional feature information, such as number (singular or plural) and verb tense. The Penn Treebank documentation (Marcus et al., 1993) defines a commonly used set of tags.

Part-of-speech tagging is an important research topic in Natural Language Processing (NLP). Taggers are often preprocessors in NLP systems, making accurate performance especially important. Much research has been done to improve tagging accuracy using several different models and methods, including: hidden Markov models (HMMs) (Kupiec, 1992), (Charniak et al., 1993); rule-based systems (Brill, 1994), (Brill, 1995); memory-based systems (Daelemans et al., 1996); maximum-entropy systems (Ratnaparkhi, 1996); path voting constraint systems (Tür and Oflazer, 1998); linear separator systems (Roth and Zelenko, 1998); and majority voting systems (van Halteren et al., 1998).

This paper describes various modifications to an HMM tagger that improve the performance to an accuracy comparable to or better than the best current single classifier taggers. This improvement comes from using second-order approximations of the Markov assumptions. Section 2 discusses a basic first-order hidden Markov model for part-of-speech tagging and extensions to that model to handle out-of-lexicon words. The new second-order HMM is described in Section 3, and Section 4 presents experimental results and conclusions.

2 Hidden Markov Models

A hidden Markov model (HMM) is a statistical construct that can be used to solve classification problems that have an inherent state sequence representation. The model can be visualized as an interlocking set of *states*. These states are connected by a set of *transition probabilities*, which indicate the probability of traveling between two given states. A process begins in some state, then at discrete time intervals, the process "moves" to a new state as dictated by the transition probabilities. In an HMM, the exact sequence of states that the process generates is unknown (i.e., *hidden*). As the process enters each state, one of a set of *output symbols* is emitted by the process. Exactly which symbol is emitted is determined by a probability distribution that is specific to each state. The output of the HMM is a sequence of output symbols.

2.1 Basic Definitions and Notation

According to (Rabiner, 1989), there are five elements needed to define an HMM:

1. N, the number of distinct states in the model. For part-of-speech tagging, N is the number of tags that can be used by the system. Each possible tag for the system corresponds to one state of the HMM.
2. M, the number of distinct output symbols in the alphabet of the HMM. For part-of-speech tagging, M is the number of words in the lexicon of the system.

3. $A = \{a_{ij}\}$, the state transition probability distribution. The probability a_{ij} is the probability that the process will move from state i to state j in one transition. For part-of-speech tagging, the states represent the tags, so a_{ij} is the probability that the model will move from tag t_i to t_j — in other words, the probability that tag t_j follows t_i. This probability can be estimated using data from a training corpus.

4. $B = \{b_j(k)\}$, the observation symbol probability distribution. The probability $b_j(k)$ is the probability that the k-th output symbol will be emitted when the model is in state j. For part-of-speech tagging, this is the probability that the word w_k will be emitted when the system is at tag t_j (i.e., $P(w_k|t_j)$). This probability can be estimated using data from a training corpus.

5. $\pi = \{\pi_i\}$, the initial state distribution. π_i is the probability that the model will start in state i. For part-of-speech tagging, this is the probability that the sentence will begin with tag t_i.

When using an HMM to perform part-of-speech tagging, the goal is to determine the most likely sequence of tags (states) that generates the words in the sentence (sequence of output symbols). In other words, given a sentence V, calculate the sequence U of tags that maximizes $P(V|U)$. The Viterbi algorithm is a common method for calculating the most likely tag sequence when using an HMM. This algorithm is explained in detail by Rabiner (1989) and will not be repeated here.

2.2 Calculating Probabilities for Unknown Words

In a standard HMM, when a word does not occur in the training data, the emit probability for the unknown word is 0.0 in the B matrix (i.e., $b_j(k) = 0.0$ if w_k is unknown). Being able to accurately tag unknown words is important, as they are frequently encountered when tagging sentences in applications. Most work in the area of unknown words and tagging deals with predicting part-of-speech information based on word endings and affixation information, as shown by work in (Mikheev, 1996), (Mikheev, 1997), (Weischedel et al., 1993), and (Thede, 1998). This section highlights a method devised for HMMs, which differs slightly from previous approaches.

To create an HMM to accurately tag unknown words, it is necessary to determine an estimate of the probability $P(w_k|t_i)$ for use in the tagger. The probability P(word contains s_j| tag is t_i) is estimated, where s_j is some "suffix" (a more appropriate term would be *word ending*, since the s_j's are not necessarily morphologically significant, but this terminology is unwieldy). This new probability is stored in a matrix $C = \{c_j(k)\}$, where $c_j(k) = P$(word has suffix s_k| tag is t_j), replaces $b_j(k)$ in the HMM calculations for unknown words. This probability can be estimated by collecting suffix information from each word in the training corpus.

In this work, suffixes of length one to four characters are considered, up to a maximum suffix length of two characters less than the length of the given word. An overall count of the number of times each suffix/tag pair appears in the training corpus is used to estimate emit probabilities for words based on their suffixes, with some exceptions. When estimating suffix probabilities, words with length four or less are not likely to contain any word-ending information that is valuable for classification, so they are ignored. Unknown words are presumed to be open-class, so words that are not tagged with an open-class tag are also ignored.

When constructing our suffix predictor, words that contain hyphens, are capitalized, or contain numeric digits are separated from the main calculations. Estimates for each of these categories are calculated separately. For example, if an unknown word is capitalized, the probability distribution estimated from capitalized words is used to predict its part of speech. However, capitalized words at the beginning of a sentence are not classified in this way—the initial capitalization is ignored. If a word is not capitalized and does not contain a hyphen or numeric digit, the general distribution is used. Finally, when predicting the possible part of speech for an unknown word, all possible matching suffixes are used with their predictions smoothed (see Section 3.2).

3 The Second-Order Model for Part-of-Speech Tagging

The model described in Section 2 is an example of a *first-order* hidden Markov model. In part-of-speech tagging, it is called a *bigram* tagger. This model works reasonably well in part-of-speech tagging, but captures a more limited

amount of the contextual information than is available. Most of the best statistical taggers use a *trigram* model, which replaces the bigram transition probability $a_{ij} = P(\tau_p = t_j | \tau_{p-1} = t_i)$ with a trigram probability $a_{ijk} = P(\tau_p = t_k | \tau_{p-1} = t_j, \tau_{p-2} = t_i)$. This section describes a new type of tagger that uses trigrams not only for the context probabilities but also for the lexical (and suffix) probabilities. We refer to this new model as a *full second-order* hidden Markov model.

3.1 Defining New Probability Distributions

The full second-order HMM uses a notation similar to a standard first-order model for the probability distributions. The A matrix contains state transition probabilities, the B matrix contains output symbol distributions, and the C matrix contains unknown word distributions. The π matrix is identical to its counterpart in the first-order model. However, the definitions of A, B, and C are modified to enable the full second-order HMM to use more contextual information to model part-of-speech tagging. In the following sections, there are assumed to be P words in the sentence with τ_p and v_p being the p-th tag and word in the sentence, respectively.

3.1.1 Contextual Probabilities

The A matrix defines the contextual probabilities for the part-of-speech tagger. As in the trigram model, instead of limiting the context to a first-order approximation, the A matrix is defined as follows:

$$A = \{a_{ijk}\}, \text{ where}$$
$$a_{ijk} = P(\tau_p = t_k | \tau_{p-1} = t_j, \tau_{p-2} = t_i), 1 \leq p \leq P$$

Thus, the transition matrix is now three dimensional, and the probability of transitioning to a new state depends not only on the current state, but also on the previous state. This allows a more realistic context-dependence for the word tags. For the boundary cases of $p = 1$ and $p = 2$, the special tag symbols NONE and SOS are used.

3.1.2 Lexical and Suffix Probabilities

The B matrix defines the lexical probabilities for the part-of-speech tagger, while the C matrix is used for unknown words. Similarly to the trigram extension to the A matrix, the approximation for the lexical and suffix probabilities can also be modified to include second-order information as follows:

$$B = \{b_{ij}(k)\} \text{ and } C = \{c_{ij}(k)\}, \text{ where}$$

$$\begin{aligned} b_{ij}(k) &= P(v_p = w_k | \tau_p = t_j, \tau_{p-1} = t_i) \\ c_{ij}(k) &= P(v_p \text{ has suffix } s_k | \tau_p = t_j, \tau_{p-1} = t_i) \\ &\quad \text{for } 1 \leq p \leq P \end{aligned}$$

In these equations, the probability of the model emitting a given word depends not only on the current state but also on the previous state. To our knowledge, this approach has not been used in tagging. SOS is again used in the $p = 1$ case.

3.2 Smoothing Issues

While the full second-order HMM is a more precise approximation of the underlying probabilities for the model, a problem can arise from sparseness of data, especially with lexical estimations. For example, the size of the B matrix is T^2W, which for the WSJ corpus is approximately 125,000,000 possible tag/tag/word combinations. In an attempt to avoid sparse data estimation problems, the probability estimates for each distribution is smoothed. There are several methods of smoothing discussed in the literature. These methods include the additive method (discussed by (Gale and Church, 1994)); the Good-Turing method (Good, 1953); the Jelinek-Mercer method (Jelinek and Mercer, 1980); and the Katz method (Katz, 1987).

These methods are all useful smoothing algorithms for a variety of applications. However, they are not appropriate for our purposes. Since we are smoothing trigram probabilities, the additive and Good-Turing methods are of limited usefulness, since neither takes into account bigram or unigram probabilities. Katz smoothing seems a little too granular to be effective in our application—the broad spectrum of possibilities is reduced to three options, depending on the number of times the given event occurs. It seems that smoothing should be based on a function of the number of occurances. Jelinek-Mercer accommodates this by smoothing the n-gram probabilities using differing coefficients (λ's) according to the number of times each n-gram occurs, but this requires holding out training data for the λ's. We have implemented a model that smooths with lower order information by using coefficients calculated from the number of occurances of each trigram, bigram, and unigram without training. This method is explained in the following sections.

3.2.1 State Transition Probabilities

To estimate the state transition probabilities, we want to use the most specific information.

However, that information may not always be available. Rather than using a fixed smoothing technique, we have developed a new method that uses variable weighting. This method attaches more weight to triples that occur more often.

The formula for the estimate $\hat{P}$ of $P(\tau_p = t_k | \tau_{p-1} = t_j, \tau_{p-2} = t_i)$ is:

$$\hat{P} = k_3 \cdot \frac{N_3}{C_2} + (1-k_3)k_2 \cdot \frac{N_2}{C_1} + (1-k_3)(1-k_2) \cdot \frac{N_1}{C_0}$$

which depends on the following numbers:

N_1	=	number of times t_k occurs
N_2	=	number of times sequence $t_j t_k$ occurs
N_3	=	number of times sequence $t_i t_j t_k$ occurs
C_0	=	total number of tags that appear
C_1	=	number of times t_j occurs
C_2	=	number of times sequence $t_i t_j$ occurs

where:

$$k_2 = \frac{\log(N_2+1)+1}{\log(N_2+1)+2}, \text{ and } k_3 = \frac{\log(N_3+1)+1}{\log(N_3+1)+2}$$

The formulas for k_2 and k_3 are chosen so that the weighting for each element in the equation for $\hat{P}$ changes based on how often that element occurs in the training data. Notice that the sum of the coefficients of the probabilities in the equation for $\hat{P}$ sum to one. This guarantees that the value returned for $\hat{P}$ is a valid probability. After this value is calculated for all tag triples, the values are normalized so that $\sum_{t_k \in T} \hat{P} = 1$, creating a valid probability distribution.

The value of this smoothing technique becomes clear when the triple in question occurs very infrequently, if at all. Consider calculating $\hat{P}$ for the tag triple *CD RB VB*. The information for this triple is:

$N_1 = 33,277$ (number of times *VB* appears)
$N_2 = 4,335$ (number of times *RB VB* appears)
$N_3 = 0$ (number of times *CD RB VB* appears)
$C_0 = 1,056,892$ (total number of tags)
$C_1 = 46,994$ (number of times *RB* appears)
$C_2 = 160$ (number of times *CD RB* appears)

Using these values, we calculate the coefficients k_2 and k_3:

$$\begin{aligned} k_2 &= \frac{\log(4,335+1)+1}{\log(4,335+1)+2} = \frac{4.637}{5.637} = 0.823 \\ k_3 &= \frac{\log(0+1)+1}{\log(0+1)+2} = \frac{1}{2} = 0.500 \end{aligned}$$

Using these values, we calculate the probability $\hat{P}$:

$$\begin{aligned} \hat{P} &= k_3 \cdot \frac{N_3}{C_2} + (1-k_3)k_2 \cdot \frac{N_2}{C_1} + (1-k_3)(1-k_2) \cdot \frac{N_1}{C_0} \\ &= 0.500 \cdot 0.000 + 0.412 \cdot 0.092 + 0.088 \cdot 0.031 \\ &= 0.041 \end{aligned}$$

If smoothing were not applied, the probability would have been 0.000, which would create problems for tagger generalization. Smoothing allows tag triples that were not encountered in the training data to be assigned a probability of occurance.

3.2.2 Lexical and Suffix Probabilities

For the lexical and suffix probabilities, we do something somewhat different than for context probabilities. Initial experiments that used a formula similar to that used for the contextual estimates performed poorly. This poor performance was traced to the fact that smoothing allowed too many words to be incorrectly tagged with tags that did not occur with that word in the training data (over-generalization). As an alternative, we calculated the smoothed probability $\hat{P}$ for words as follows:

$$\hat{P} = \left(\frac{\log(N_3+1)+1}{\log(N_3+1)+2}\right)\frac{N_3}{C_2} + \left(\frac{1}{\log(N_3+1)+2}\right)\frac{N_2}{C_1}$$

where:

N_2	=	number of times word w_k occurs with tag t_j
N_3	=	number of times word w_k occurs with tag t_j preceded by tag t_i
C_1	=	number of times t_j occurs
C_2	=	number of times sequence $t_i t_j$ occurs

Notice that this method assigns a probability of 0.0 to a word/tag pair that does not appear in the training data. This prevents the tagger from trying every possible combination of word and tag, something which both increases running time and decreases the accuracy. We believe the low accuracy of the original smoothing scheme emerges from the fact that smoothing the lexical probabilities too far allows the contextual information to dominate at the expense of the lexical information. A better smoothing approach for lexical information could possibly be created by using some sort of word class idea, such as the genotype idea used in (Tzoukermann and Radev, 1996), to improve our $\hat{P}$ estimate.

In addition to choosing the above approach for smoothing the C matrix for unknown words, there is an additional issue of choosing which suffix to use when predicting the part of speech. There are many possible answers, some of which are considered by (Thede, 1998): use the longest matching suffix, use an entropy measure to determine the "best" affix to use, or use an average. A voting technique for $c_{ij}(k)$ was determined that is similar to that used for contextual smoothing but is based on different length suffixes.

Let s_4 be the length four suffix of the given word. Define s_3, s_2, and s_1 to be the length three, two, and one suffixes respectively. If the length of the word is six or more, these four suffixes are used. Otherwise, suffixes up to length $n-2$ are used, where n is the length of the word. Determine the longest suffix of these that matches a suffix in the training data, and calculate the new smoothed probability:

$$\hat{P}_{ij}(s_k) = \begin{cases} f(N_k)\hat{c}_{ij}(s_k) + (1 - f(N_k))\hat{P}_{ij}(s_{k-1}), & 1 < k \leq 4 \\ \hat{c}_{ij}(s_1), & k = 1 \end{cases}$$

where:

- $f(x) = \frac{\log(x+1)+1}{\log(x+1)+2}$
- N_k = the number of times the suffix s_k occurs in the training data.
- $\hat{c}_{ij}(s_k)$ = the estimate of $c_{ij}(s_k)$ from the previous lexical smoothing.

After calculating $\hat{P}$, it is normalized. Thus, suffixes of length four are given the most weight, and a suffix receives more weight the more times it appears. Information provided by suffixes of length one to four are used in estimating the probabilities, however.

3.3 The New Viterbi Algorithm

Modification of the lexical and contextual probabilities is only the first step in defining a full second-order HMM. These probabilities must also be combined to select the most likely sequence of tags that generated the sentence. This requires modification of the Viterbi algorithm. First, the variables δ and ψ from (Rabiner, 1989) are redefined, as shown in Figure 1. These new definitions take into account the added dependencies of the distributions of A, B, and C. We can then calculate the most likely tag sequence using the modification of the Viterbi algorithm shown in Figure 1. The running time of this algorithm is $O(NT^3)$, where N is the length of the sentence, and T is the number of tags. This is asymptotically equivalent to the running time of a standard trigram tagger that maximizes the probability of the entire tag sequence.

4 Experiment and Conclusions

The new tagging model is tested in several different ways. The basic experimental technique is a 10-fold cross validation. The corpus in question is randomly split into ten sections with nine of the sections combined to train the tagger and the tenth for testing. The results of the ten possible training/testing combinations are merged to give an overall accuracy measure. The tagger was tested on two corpora—the Brown corpus (from the Treebank II CD-ROM (Marcus et al., 1993)) and the Wall Street Journal corpus (from the same source). Comparing results for taggers can be difficult, especially across different researchers. Care has been taken in this paper that, when comparing two systems, the comparisons are from experiments that were as similar as possible and that differences are highlighted in the comparison.

First, we compare the results on each corpus of four different versions of our HMM tagger: a standard (bigram) HMM tagger, an HMM using second-order lexical probabilities, an HMM using second-order contextual probabilities (a standard trigram tagger), and a full second-order HMM tagger. The results from both corpora for each tagger are given in Table 1. As might be expected, the full second-order HMM had the highest accuracy levels. The model using only second-order contextual information (a standard trigram model) was second best, the model using only second-order lexical information was third, and the standard bigram HMM had the lowest accuracies. The full second-order HMM reduced the number of errors on known words by around 16% over bigram taggers (raising the accuracy about 0.6-0.7%), and by around 6% over conventional trigram taggers (accuracy increase of about 0.2%). Similar results were seen in the overall accuracies. Unknown word accuracy rates were increased by around 2-3% over bigrams.

The full second-order HMM tagger is also compared to other researcher's taggers in Table 2. It is important to note that both SNOW, a linear separator model (Roth and Zelenko,

THE SECOND-ORDER VITERBI ALGORITHM

The variables:

- $\delta_p(i,j) = \max_{\tau_1,\ldots,\tau_{p-2}} P(\tau_1,\ldots,\tau_{p-2},\tau_{p-1}=t_i,\tau_p=t_j,v_1,\ldots v_p), 2 \le p \le P$
- $\psi_p(i,j) = \arg\max_{\tau_1,\ldots,\tau_{p-2}} P(\tau_1,\ldots,\tau_{p-2},\tau_{p-1}=t_i,\tau_p=t_j,v_1,\ldots v_p), 2 \le p \le P$

The procedure:

1. $\delta_1(i,j) = \left\{ \begin{array}{ll} \pi_i b_{ij}(v_1), & \text{if } v_1 \text{ is known} \\ \pi_i c_{ij}(v_1), & \text{if } v_1 \text{ is unknown} \end{array} \right\}, 1 \le i,j \le N$

   $\psi_1(i,j) = 0, 1 \le i,j \le N$

2. $\delta_p(j,k) = \left\{ \begin{array}{ll} \max_{1\le i\le N}[\delta_{p-1}(i,j)a_{ijk}]b_{jk}(v_p), & \text{if } v_p \text{ is known} \\ \max_{1\le i\le N}[\delta_{p-1}(i,j)a_{ijk}]c_{jk}(v_p), & \text{if } v_p \text{ is unknown} \end{array} \right\}, 1 \le i,j,k \le N, 2 \le p \le P$

   $\psi_p(j,k) = \arg\max_{1\le i\le N}[\delta_{p-1}(i,j)a_{ijk}], 1 \le i,j,k \le N, 2 \le p \le P$

3. $P^* = \max_{1\le i,j\le N} \delta_P(i,j)$

   $\tau_P^* = \arg_j \max_{1\le i,j\le N} \delta_P(i,j)$

   $\tau_{P-1}^* = \arg_i \max_{1\le i,j\le N} \delta_P(i,j)$

4. $\tau_p^* = \psi_{p+1}(\tau_{p+1}^*, \tau_{p+2}^*), p = P-2, P-3, \ldots, 2, 1$

Figure 1: Second-Order Viterbi Algorithm

Comparison on Brown Corpus			
Tagger Type	**Known**	**Unknown**	**Overall**
Standard Bigram	95.94%	80.61%	95.60%
Second-Order Lexical only	96.23%	81.42%	95.90%
Second-Order Contextual only	96.41%	82.69%	96.11%
Full Second-Order HMM	96.62%	83.46%	96.33%

Comparison on WSJ Corpus			
Tagger Type	**Known**	**Unknown**	**Overall**
Standard Bigram	96.52%	82.40%	96.25%
Second-Order Lexical only	96.80%	83.63%	96.54%
Second-Order Contextual only	96.90%	84.10%	96.65%
Full Second-Order HMM	97.09%	84.88%	96.86%

% Error Reduction of Second-Order HMM		
System Type Compared	**Brown**	**WSJ**
Bigram	16.6%	16.3%
Lexical Trigrams Only	10.5%	9.2%
Contextual Trigrams Only	5.7%	6.3%

Table 1: Comparison between Taggers on the Brown and WSJ Corpora

1998), and the voting constraint tagger (Tür and Oflazer, 1998) used training data that contained full lexical information (i.e., no unknown words), as well as training and testing data that did not cover the entire WSJ corpus. This use of a full lexicon may have increased their accuracy beyond what it would have been if the model were tested with unknown words. The standard trigram tagger data is from (Weischedel et al., 1993). The MBT (Daelemans et al., 1996)

Tagger Type	Known	Unknown	Overall	Open/Closed Lexicon?	Testing Method
Standard Trigram (Weischedel et al., 1993)	96.7%	85.0%	**96.3%**	open	full WSJ [1]
MBT (Daelemans et al., 1996)	96.7%	90.6% [2]	**96.4%**	open	fixed WSJ cross-validation
Rule-based (Brill, 1994)	——	82.2%	**96.6%**	open	fixed full WSJ [3]
Maximum-Entropy (Ratnaparkhi, 1996)	——	85.6%	**96.6%**	open	fixed full WSJ [3]
Full Second-Order HMM	**97.1%**	**84.9%**	**96.9%**	**open**	**full WSJ cross-validation**
SNOW (Roth and Zelenko, 1998)	97.2%	——	——	closed	fixed subset of WSJ [4]
Voting Constraints (Tür and Oflazer, 1998)	97.5%	——	——	closed	subset of WSJ cross-validation [5]
Full Second-Order HMM	**98.05%**	——	——	**closed**	**full WSJ cross-validation**

Table 2: Comparison between Full Second-Order HMM and Other Taggers

did not include numbers in the lexicon, which accounts for the inflated accuracy on unknown words. Table 2 compares the accuracies of the taggers on known words, unknown words, and overall accuracy. The table also contains two additional pieces of information. The first indicates if the corresponding tagger was tested using a *closed* lexicon (one in which all words appearing in the testing data are known to the tagger) or an *open* lexicon (not all words are known to the system). The second indicates whether a hold-out method (such as cross-validation) was used, and whether the tagger was tested on the entire WSJ corpus or a reduced corpus.

Two cross-validation tests with the full second-order HMM were run: the first with an open lexicon (created from the training data), and the second where the entire WSJ lexicon was used for each test set. These two tests allow more direct comparisons between our system and the others. As shown in the table, the full second-order HMM has improved overall accuracies on the WSJ corpus to state-of-the-art levels—96.9% is the greatest accuracy reported on the full WSJ for an experiment using an open lexicon. Finally, using a closed lexicon, the full second-order HMM achieved an accuracy of 98.05%, the highest reported for the WSJ corpus for this type of experiment.

The accuracy of our system on unknown words is 84.9%. This accuracy was achieved by creating separate classifiers for capitalized, hyphenated, and numeric digit words: tests on the Wall Street Journal corpus with the full second-order HMM show that the accuracy rate on unknown words without separating these types of words is only 80.2%.[6] This is below the performance of our bigram tagger that separates the classifiers. Unfortunately, unknown word accuracy is still below some of the other systems. This may be due in part to experimental differences. It should also be noted that some of these other systems use hand-crafted rules for unknown word rules, whereas our system uses only statistical data. Adding additional rules to our system could result in comparable performance. Improving our model on unknown words is a major focus of future research.

In conclusion, a new statistical model, the full second-order HMM, has been shown to improve part-of-speech tagging accuracies over current models. This model makes use of second-order approximations for a hidden Markov model and

[1] The full WSJ is used, but the paper does not indicate whether a cross-validation was performed.

[2] MBT did not place numbers in the lexicon, so all numbers were treated as unknown words.

[3] Both the rule-based and maximum-entropy models use the full WSJ for training/testing with only a single test set.

[4] SNOW used a fixed subset of WSJ for training and testing with no cross-validation.

[5] The voting constraints tagger used a subset of WSJ for training and testing with cross-validation.

[6] Mikheev (1997) also separates suffix probabilities into different estimates, but fails to provide any data illustrating the implied accuracy increase.

improves the state of the art for taggers with no increase in asymptotic running time over traditional trigram taggers based on the hidden Markov model. A new smoothing method is also explained, which allows the use of second-order statistics while avoiding sparse data problems.

References

Eric Brill. 1994. A report of recent progress in transformation-based error-driven learning. *Proceedings of the Twelfth National Conference on Artifical Intelligence*, pages 722–727.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for part-of-speech tagging. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789.

Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. 1996. MBT: A memory-based part of speech tagger-generator. *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27.

William A. Gale and Kenneth W. Church. 1994. What's wrong with adding one? In *Corpus-Based Research into Language*. Rodolpi, Amsterdam.

I. J. Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. *Proceedings of the Workshop on Pattern Recognition in Practice.*

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

Julian Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6(3):225–242.

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

Andrei Mikheev. 1996. Unsupervised learning of word-category guessing rules. *Proceedings of the 34th Annual Meeting of the Association for Compuatational Linguistics*, pages 327–334.

Andrei Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Computational Linguistics*, 23(3):405–423.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceeding of the IEEE*, pages 257–286.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142.

Dan Roth and Dmitry Zelenko. 1998. Part of speech tagging using a network of linear separators. *Proceedings of COLING-ACL '98*, pages 1136–1142.

Scott M. Thede. 1998. Predicting part-of-speech information about unknown words using statistical methods. *Proceedings of COLING-ACL '98*, pages 1505–1507.

Gökhan Tür and Kemal Oflazer. 1998. Tagging English by path voting constraints. *Proceedings of COLING-ACL '98*, pages 1277–1281.

Evelyne Tzoukermann and Dragomir R. Radev. 1996. Using word class for part-of-speech disambiguation. *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 1–13.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 1998. Improving data driven wordclass tagging by system combination. *Proceedings of COLING-ACL '98*, pages 491–497.

Ralph Weischedel, Marie Meeter, Richard Schwartz, Lance Ramshaw, and Jeff Palmucci. 1993. Coping with ambiguity and unknown words through probabilitic models. *Computational Linguistics*, 19:359–382.