Mandarin Part-of-Speech Tagging and Discriminative Reranking

Zhongqiang Huang[1]
[1]Purdue University
West Lafayette, IN 47907
zqhuang@purdue.edu

Mary P. Harper[1,2]
[2]University of Maryland
College Park, MD 20742
mharper@casl.umd.edu

Wen Wang
SRI International
Menlo Park, CA 94025
wwang@speech.sri.com

Abstract

We present in this paper methods to improve HMM-based part-of-speech (POS) tagging of Mandarin. We model the emission probability of an unknown word using all the characters in the word, and enrich the standard left-to-right trigram estimation of word emission probabilities with a right-to-left prediction of the word by making use of the current and next tags. In addition, we utilize the RankBoost-based reranking algorithm to rerank the N-best outputs of the HMM-based tagger using various n-gram, morphological, and dependency features. Two methods are proposed to improve the generalization performance of the reranking algorithm. Our reranking model achieves an accuracy of 94.68% using n-gram and morphological features on the Penn Chinese Treebank 5.2, and is able to further improve the accuracy to 95.11% with the addition of dependency features.

1 Introduction

Part-of-speech (POS) tagging is potentially helpful for many advanced natural language processing tasks, for example, named entity recognition, parsing, and sentence boundary detection. Much research has been done to improve tagging performance for a variety of languages. The state-of-the-art systems have achieved an accuracy of 97% for English on the Wall Street Journal (WSJ) corpus (which contains 4.5M words) using various models (Brants, 2000; Ratnaparkhi, 1996; Thede and Harper, 1999). Lower accuracies have been reported in the literature for Mandarin POS tagging (Tseng et al., 2005; Xue et al., 2002). This is, in part, due to the relatively small size and the different annotation guidelines (e.g., granularity of the tag set) for the annotated corpus of Mandarin. Xue at el. (2002) and Tseng at el. (2005) reported accuracies of 93% and 93.74% on CTB-I (Xue et al., 2002) (100K words) and CTB 5.0 (500K words), respectively, each using a Maximum Entropy approach. The characteristics of Mandarin make it harder to tag than English. Chinese words tend to have greater POS tag ambiguity than English. Tseng at el. (2005) reported that 29.9% of the words in CTB have more than one POS assignment compared to 19.8% of the English words in WSJ. Moreover, the morphological properties of Chinese words complicate the prediction of POS type for unknown words.

These challenges for Mandarin POS tagging suggest the need to develop more sophisticated methods. In this paper, we investigate the use of a discriminative reranking approach to increase Mandarin tagging accuracy. Reranking approaches (Charniak and Johnson, 2005; Chen et al., 2002; Collins and Koo, 2005; Ji et al., 2006; Roark et al., 2006) have been successfully applied to many NLP applications, including parsing, named entity recognition, sentence boundary detection, etc. To the best of our knowledge, reranking approaches have not been used for POS tagging, possibly due to the already high levels of accuracy for English, which leave little room for further improvement. However, the relatively poorer performance of existing methods on Mandarin POS tagging makes reranking a much more compelling technique to evaluate. In this paper, we use reranking to improve tagging performance of an HMM tagger adapted to

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1093–1102, Prague, June 2007.

Mandarin. Hidden Markov models are simple and effective, but unlike discriminative models, such as Maximum Entropy models (Ratnaparkhi, 1996) and Conditional Random Fields (John Lafferty, 2001), they have more difficulty utilizing a rich set of conditionally dependent features. This limitation can be overcome by utilizing reranking approaches, which are able to make use of the features extracted from the tagging hypotheses produced by the HMM tagger. Reranking also has advantages over MaxEnt and CRF models. It is able to use any features extracted from entire labeled sentences, including those that cannot be incorporated into MaxEnt and CRF models due to inference difficulties. In addition, reranking methods are able to utilize the information provided by N-best lists. Finally, the decoding phase of reranking is much simpler.

The rest of the paper is organized as follows. We describe the HMM tagger in Section 2. We discuss the modifications to better handle unknown words in Mandarin and to enrich the word emission probabilities through the combination of bi-directional estimations. In Section 3, we first describe the reranking algorithm and then propose two methods to improve its performance. We also describe the features that will be used for Mandarin POS reranking in Section 3. Experimental results are given in Section 4. Conclusions and future work appear in Section 5.

2 The HMM Model

2.1 Porting English Tagger to Mandarin

The HMM tagger used in this work is a second-order HMM tagger initially developed for English by Thede and Harper (1999). This state-of-the-art second-order HMM tagger uses trigram transition probability estimations, $P(t_i|t_{i-2}t_{i-1})$, and trigram emission probability estimations, $P(w_i|t_{i-1}t_i)$. Let t_1^i denote the tag sequence $t_1, \cdots, t_i$, and w_1^i denote the word sequence $w_1, \cdots, w_i$. The tagging problem can be formally defined as finding the best tag sequence $\tau(w_1^N)$ for the word sequence w_1^N of length N as follows[1]:

$$\begin{aligned}
\tau(w_1^N) &= \arg\max_{t_1^N} P(t_1^N|w_1^N) = \arg\max_{t_1^N} \frac{P(t_1^N w_1^N)}{P(w_1^N)} \\
&= \arg\max_{t_1^N} P(t_1^N w_1^N) \qquad (1) \\
&= \arg\max_{t_1^N} \prod_i P(t_i|t_1^{i-1} w_1^{i-1}) P(w_i|t_1^i w_1^{i-1}) \\
&\approx \arg\max_{t_1^N} \prod_i P(t_i|t_{i-2}t_{i-1}) P(w_i|t_{i-1}t_i) \qquad (2)
\end{aligned}$$

[1]We assume that symbols exist implicitly for boundary conditions.

The best tag sequence $\tau(w_1^N)$ can be determined efficiently using the Viterbi algorithm.

For estimating emission probabilities of unknown words (i.e., words that do not appear in the training data) in English (and similarly for other inflected languages), a weighted sum of $P(s_i^k|t_{i-1}t_i)$ (with k up to four) was used as an approximation, where s_i^k is the suffix of length k of word w_i (e.g., s_i^1 is the last character of word w_i). The suffix information and three binary features (i.e., whether the word is capitalized, whether the word is hyphenated, and whether the word contains numbers) are combined to estimate the emission probabilities of unknown words.

The interpolation weights for smoothing transition, emission, and suffix probabilities were estimated using the log-based Thede smoothing method (Thede and Harper, 1999) as follows:

$$\begin{aligned}
&P_{Thede}(\text{n-gram}) \\
&\quad = \lambda(\text{n-gram}) P_{ML}(\text{n-gram}) + \\
&\qquad (1 - \lambda(\text{n-gram})) P_{Thede}(\text{(n-1)-gram})
\end{aligned}$$

where:

$$\begin{aligned}
P_{ML}(\text{n-gram}) &= \text{the ML estimation} \\
\lambda(\text{n-gram}) &= f(\text{n-gram count}) \\
f(x) &= \frac{\log_a(x+1) + b}{\log_a(x+1) + (b+1)}
\end{aligned}$$

While porting the HMM-based English POS tagger to Mandarin is fairly straightforward for words seen in the training data, some thought is required to handle unknown words due to the morphology differences between the two languages. First, in Mandarin, there is no capitalization and no hyphenation. Second, although Chinese has morphology, it is not the same as in English; words tend to contain far fewer characters than inflected words in English, so word endings will tend to be short, say one or two characters long. Hence, in our baseline model (denoted *HMM baseline*), we simply utilize word endings of up to two characters in length along with a binary feature of whether the word contains numbers or not. In the next two subsections, we describe two ways in which we enhance this simple HMM baseline model.

2.2 Improving the Mandarin Unknown Word Model

Chinese words are quite different from English words, and the word formation process for Chinese words can be quite complex (Packard, 2000). Indeed, the last characters in a Chinese word are, in some cases, most informative of the POS type, while for others, it is the characters at the beginning. Furthermore, it is not uncommon for a character in the middle of a word to provide some evidence for the POS type of the word. Hence, we chose to employ a rather simple but effective method to estimate the emission probability, $P(w_i|t_{i-1}, t_i)$, of an unknown word, w_i. We use the geometric average[2] of the emission probability of the characters in the word, i.e., $P(c_k|t_{i-1}, t_i)$ with c_k being the k-th character in the word. Since some of the characters in w_i may not have appeared in any word tagged as t_i in that context in the training data, only characters that are observed in this context are used in the computation of the geometric average, as shown below:

$$P(w_i|t_{i-1}, t_i) = \sqrt[n]{\prod_{c_k \in w_i, P(c_k|t_{i-1}, t_i) \neq 0} P(c_k|t_{i-1}, t_i)} \quad (3)$$

where

$$n = |\{c_k \in w_i | P(c_k|t_{i-1}, t_i) \neq 0\}|$$

2.3 Bi-directional Word Probability Estimation

In Equation 2, the word emission probability $P(w_i|t_{i-1}t_i)$ is a left-to-right prediction that depends on the current tag t_i associated with w_i, as well as its previous tag t_{i-1}. Although the interaction between w_i and the next tag t_{i+1} is captured to some extent when t_{i+1} is generated by the model, this implicit interaction may not be as effective as adding the information more directly to the model. Hence, we chose to apply the constraint explicitly in our HMM framework by replacing $P(w_i|t_{i-1}t_i)$ in Equation 2 with $P^{\lambda}(w_i|t_{i-1}t_i)P^{1-\lambda}(w_i|t_i t_{i+1})$ for both known and unknown words, with $\tau(w_1^N)$ determined by:

$$\begin{aligned} \tau(w_1^N) &= \arg\max_{t_1^N} \prod_i (P(t_i|t_{i-2}t_{i-1}) \times \\ &\quad P^{\lambda}(w_i|t_{i-1}t_i)P^{1-\lambda}(w_i|t_i t_{i+1})) \quad (4) \end{aligned}$$

This corresponds to a mixture model of two generation paths, one from the left and one from the right, to approximate $\tau(w_1^N)$ in Equation 1 in a different way.

$$\begin{aligned} \tau(w_1^N) &= \arg\max_{t_1^N} P(t_1^N w_1^N) \\ &= \arg\max_{t_1^N} P(t_1^N)P(w_1^N|t_1^N) \\ P(t_1^N) &\approx \prod_i P(t_i|t_{i-1}t_{i-2}) \\ P(w_1^N|t_1^N) &= P^{\lambda}(w_1^N|t_1^N)P^{1-\lambda}(w_1^N|t_1^N) \\ &\approx \prod_i P^{\lambda}(w_i|t_{i-1}t_i)P^{1-\lambda}(w_i|t_i t_{i+1}) \end{aligned}$$

In this case, the decoding process involves the computation of three local probabilities, i.e., $P(t_i|t_{i-2}t_{i-1})$, $P(w_i|t_{i-1}t_i)$, and $P(w_i|t_i t_{i+1})$. By using a simple manipulation that shifts the time index of $P(w_i|t_i t_{i+1})$ in Equation 4 by two time slices[3] (i.e., by replacing $P(w_i|t_i t_{i+1})$ with $P(w_{i-2}|t_{i-2}t_{i-1})$), we are able to compute $\tau(w_1^N)$ in Equation 4 with the same asymptotic time complexity of decoding as in Equation 2.

3 Discriminative Reranking

In this section, we describe our use of the RankBoost-based (Freund and Schapire, 1997; Freund et al., 1998) discriminative reranking approach that was originally developed by Collins and Koo (2005) for parsing. It provides an additional avenue for improving tagging accuracy, and also allows us to investigate the impact of various features on Mandarin tagging performance. The reranking algorithm takes as input a list of candidates produced by some probabilistic model, in our case the HMM tagger, and reranks these candidates based on a set of features. We first introduce Collins' reranking algorithm in Subsection 3.1, and then describe two modifications in Subsections 3.2 and 3.3 that were designed to improve the generalization performance of the reranking algorithm for our POS tagging task. The reranking features that are used for POS tagging are then described in Subsection 3.4.

3.1 Collins' Reranking Algorithm

For training the reranker for the POS tagging task, there are n sentences $\{s_i : i = 1, \cdots, n\}$ each with n_i candidates $\{x_{i,j} : j = 1, \cdots, n_i\}$ along with

[2] Based on preliminary testing, the geometric average provided greater tag accuracy than the arithmetic average.

[3] Replacing $P(w_i|t_i t_{i+1})$ with $P(w_{i-1}|t_{i-1}t_i)$ also gives the same solution.

the log-probability $L(x_{i,j})$ produced by the HMM tagger. Each tagging candidate $x_{i,j}$ in the training data has a "goodness" score $Score(x_{i,j})$ that measures the similarity between the candidate and the gold reference. For tagging, we use tag accuracy as the similarity measure. Without loss of generality, we assume that $x_{i,1}$ has the highest score, i.e., $Score(x_{i,1}) \geq Score(x_{i,j})$ for $j = 2, \cdots, n_i$. To summarize, the training data consists of a set of examples $\{x_{i,j} : i = 1, \cdots, n; j = 1, \cdots, n_i\}$, each along with a "goodness" score $Score(x_{i,j})$ and a log-probability $L(x_{i,j})$.

A set of indicator functions $\{h_k : k = 1, \cdots, m\}$ are used to extract binary features $\{h_k(x_{i,j}) : k = 1, \cdots, m\}$ on each example $x_{i,j}$. An example of an indicator function for POS tagging is given below:

$$
\begin{aligned}
h_{2143}(x) \quad = \quad & 1 \text{ if } x \text{ contains n-gram "go/VV to"} \\
& 0 \text{ otherwise}
\end{aligned}
$$

Each indicator function h_k is associated with a weight parameter α_k which is real valued. In addition, a weight parameter α_0 is associated with the log-probability $L(x_{i,j})$. The ranking function of candidate $x_{i,j}$ is defined as $\alpha_0 L(x_{i,j}) + \sum_{k=1}^{m} \alpha_k h_k(x_{i,j})$.

The objective of the training process is to set the parameters $\bar{\alpha} = \{\alpha_0, \alpha_1, \cdots, \alpha_m\}$ to minimize the following loss function Loss$(\bar{\alpha})$ (which is an upper bound on the training error):

$$
\text{Loss}(\bar{\alpha}) = \sum_i \sum_{j=2}^{n_i} S_{i,j} e^{-M_{i,j}(\bar{\alpha})}
$$

where $S_{i,j}$ is the weight function that gives the importance of each example, and $M_{i,j}(\bar{\alpha})$ is the margin:

$$
\begin{aligned}
S_{i,j} &= Score(x_{i,1}) - Score(x_{i,j}) \\
M_{i,j}(\bar{\alpha}) &= \alpha_0(L(x_{i,1}) - L(x_{i,j})) + \\
& \quad \sum_{k=1}^{m} \alpha_k (h_k(x_{i,1}) - h_k(x_{i,j}))
\end{aligned}
$$

All of the α_i's are initially set to zero. The value of α_0 is determined first to minimize the loss function and is kept fixed afterwards. Then a greedy sequential [4] optimization method is used in each iteration (i.e., a boosting round) to select the feature that has the most impact on reducing the loss function and then update its weight parameter accordingly. For each $k \in \{1, \cdots, m\}$, $(h_k(x_{i,1}) - h_k(x_{i,j}))$ can only take one of the three values: +1, -1, or 0. Thus the training examples can be divided into three subsets with respect to k:

$$
\begin{aligned}
A_k^+ &= \{(i,j) : (h_k(x_{i,1}) - h_k(x_{i,j})) = +1\} \\
A_k^- &= \{(i,j) : (h_k(x_{i,1}) - h_k(x_{i,j})) = -1\} \\
A_k^0 &= \{(i,j) : (h_k(x_{i,1}) - h_k(x_{i,j})) = 0\}
\end{aligned}
$$

The new loss after adding the update parameter δ to the parameter α_k is shown below:

$$
\begin{aligned}
\text{Loss}(\bar{\alpha}, k, \delta) &= \sum_{(i,j) \in A_k^+} S_{i,j} e^{-M_{i,j}(\bar{\alpha}) - \delta} + \\
& \quad \sum_{(i,j) \in A_k^-} S_{i,j} e^{-M_{i,j}(\bar{\alpha}) + \delta} + \\
& \quad \sum_{(i,j) \in A_k^0} S_{i,j} e^{-M_{i,j}(\bar{\alpha})} \\
&= e^{-\delta} W_k^+ + e^{\delta} W_k^- + W_k^0
\end{aligned}
$$

The best feature/update pair (k^*, δ^*) that minimizes Loss$(\bar{\alpha}, k, \delta)$ is determined using the following formulas:

$$
k^* = \arg\max_k \left| \sqrt{W_k^+} - \sqrt{W_k^-} \right| \tag{5}
$$

$$
\delta^* = \frac{1}{2} \log \frac{W_{k^*}^+}{W_{k^*}^-} \tag{6}
$$

The update formula in Equation 6 is problematic when either $W_{k^*}^+$ or $W_{k^*}^-$ is zero. W_k^+ is zero if h_k never takes on a value 1 for any $x_{i,1}$ with value 0 on a corresponding $x_{i,j}$ for $j = 2, \cdots, n_i$ (and similarly for W_k^-). Collins introduced a smoothing parameter ϵ to address this problem, resulting in a slight modification to the update formula:

$$
\delta^* = \frac{1}{2} \log \frac{W_{k^*}^+ + \epsilon Z}{W_{k^*}^- + \epsilon Z} \tag{7}
$$

The value of ϵ plays an important role in this formula. If ϵ is set too small, the smoothing factor ϵZ would not prevent setting δ^* to a potentially overly large absolute value, resulting in over-fitting. If ϵ is set too large, then the opposite condition of under-training could result. The value of ϵ is determined based on a development set.

[4] Parallel optimization algorithms exist and have comparable performance according to (Collins et al., 2002).

3.2 Update Once

Collins' method allows multiple updates to the weight of a feature based on Equations 5 and 7. We found that for those features for which either W_k^+ or W_k^- equals zero, the update formula in Equation 7 can only increase their weight (in absolute value) in one direction. Although these features are strong and useful, setting their weights too large can be undesirable in that it limits the use of other features for reducing the loss.

Based on this analysis, we have developed and evaluated an *update-once* method, in which we use the update formula in Equation 7 but limit weight updates so that once a feature is selected on a certain iteration and its weight parameter is updated, it cannot be updated again. Using this method, the weights of the strong features are not allowed to prevent additional features from being considered during the training phase.

3.3 Regularized Reranking

Although the update-once method may attenuate over-fitting to some extent, it also prevents adjusting the value of any weight parameter that is initially set too high or too low in an earlier boosting round. In order to design a more sophisticated weight update method that allows multiple updates in both directions while penalizing overly large weights, we have also investigated the addition of a regularization term $R(\bar{\alpha})$, an exponential function of $\bar{\alpha}$, to the loss function:

$$\begin{aligned}\text{RegLoss}(\bar{\alpha}) &= \sum_i \sum_{j=2}^{n_i} S_{i,j} e^{-M_{i,j}(\bar{\alpha})} + R(\bar{\alpha}) \\ R(\bar{\alpha}) &= \sum_{k=1}^{m} p_k \cdot (e^{-\alpha_k} + e^{\alpha_k} - 2)\end{aligned}$$

where p_k is the penalty weight of parameter α_k. The reason that we chose this form of regularization is that $(e^{-\alpha_k} + e^{\alpha_k} - 2)$ is a symmetric, monotonically decreasing function of $|\alpha_k|$, and more importantly it provides a closed analytical expression of the weight update formula similar to Equations 5 and 6. Hence, the best feature/update pair for the regularized loss function is defined as follows:

$$\begin{aligned}k^* &= \arg\max_k \left| \sqrt{W_k^+ + p_k e^{-\alpha_k}} - \sqrt{W_k^- + p_k e^{+\alpha_k}} \right| \\ \delta^* &= \frac{1}{2} \log \frac{W_{k^*}^+ + p_{k^*} e^{-\alpha_{k^*}}}{W_{k^*}^- + p_{k^*} e^{+\alpha_{k^*}}}\end{aligned}$$

There are many ways of choosing p_k, the penalty weight of α_k. In this paper, we use the values of $\beta \cdot (W_k^+ + W_k^-)$ at the beginning of the first iteration (after α_0 is determined) for p_k, where β is a weighting parameter to be tuned on the development set. The regularized weight update formula has many advantages. It is always well defined no matter what value W_k^+ and W_k^- take, in contrast to Equation 6. For all features, even in the case when either W_k^+ or W_k^- equals zero, the regularized update formula allows weight updates in two directions. If the weight is small, W_k^+ and W_k^- have more impact on determining the weight update direction, however, when the weight becomes large, the regularization factors $p_k e^{-\alpha}$ and $p_k e^{+\alpha}$ favor reducing the weight.

3.4 Reranking Features

A reranking model has the flexibility of incorporating any type of feature extracted from N-best candidates. For the work presented in this paper, we examine three types of features. For each window of three word/tag pairs, we extract all the n-grams, except those that are comprised of only one word/tag pair, or only tags, or only words, or do not include either the word or tag in the center word/tag pair. These constitute the *n-gram* feature set.

In order to better handle unknown words, we also extract the two most important types of *morphological* features[5] that were utilized in (Tseng et al., 2005) for those words that appear no more than seven times (following their convention) in the training set:

Affixation features: we use character n-gram prefixes and suffixes for n up to 4. For example, for word/tag pair 资料袋/NN (Information-Bag, i.e., folder), we add the following features: (prefix1, 资, NN), (prefix2, 资料, NN), (prefix3, 资料袋, NN), (suffix1, 袋, NN), (suffix2, 料袋, NN), (suffix3, 资料袋, NN).

AffixPOS features[6]: we used the training set to build a prefix/POS and suffix/POS dictionary associating possible tags with each prefix and

[5] Tseng at el. also used other morphological features that require additional resources to which we do not have access.

[6] AffixPOS features are somewhat different from the CTB-Morph features used in (Tseng et al., 2005), where a morpheme/POS dictionary with the possible tags for all morphemes in the training set was used instead of two separate dictionaries for prefix and suffix. AffixPOS features perform slightly better in our task than the CTB-morph features.

suffix in the training set. The AffixPOS features indicate the set of tags a given affix could have. For the same example 资料袋/NN, 资 occurred as prefix in both NN and VV words in the training data. So we add the following features based on the prefix 资: (prefix, 资, NN, 1, NN), (prefix, 资, VV, 1, NN), and (prefix, 资, *X*, 0, NN) for every tag *X* not in {NN, VV}, where 1 and 0 are indicator values. Features are extracted in the similar way for the suffix 袋.

The n-gram and morphological features are easy to compute, however, they have difficulty in capturing the long distance information related to syntactic relationships that might help POS tagging accuracy. In order to examine the effectiveness of utilizing syntactic information in tagging, we have also experimented with *dependency* features that are extracted based on automatic parse trees. First a bracketing parser (the Charniak parser (Charniak, 2000) in our case) is used to generate the parse tree of a sentence, then the *const2dep* tool developed by Hwa was utilized to convert the bracketing tree to a dependency tree based on the head percolation table developed by the second author. The dependency tree is comprised of a set of dependency relations among word pairs. A dependency relation is a triple ⟨*word-a*, *word-b*, *relation*⟩, in which *word-a* is governed by *word-b* with grammatical relation denoted as *relation*. For example, in the sentence "西藏(Tibet) 经济(economy) 建设(construction) 取得(achieves) 显著(significant) 成绩(accomplishments)", one example dependency relation is ⟨取得, 成绩, *mod*⟩. Given these dependency relations, we then extract dependency features (in total 36 features for each relation) by examining the POS tags of the words for each tagging candidate of a sentence. The relative positions of the word pairs are also taken into account for some features. For example, if 取得 and 成绩 in the above sentence are tagged as VV and NN respectively in one candidate, then two example dependency features are (dep-1, 取得, VV, 成绩, NN, *mod*), (dep-14, 取得, VV, NN, *right*, *mod*), in which dep-1 and dep-14 are feature types and *right* indicates that *word-b* (取得) is to the right of *word-a* (成绩).

4 Experiments

4.1 Data

The most recently released Penn Chinese Treebank 5.2 (denoted CTB, released by LDC) is used in our experiments. It contains 500K words, 800K characters, 18K sentences, and 900 data files, including articles from the Xinhua news agency (China-Mainland), Information Services Department of HKSAR (Hongkong), and Sinorama magazine (Taiwan). Its format is similar to the English WSJ Penn Treebank, and it was carefully annotated. There are 33 POS tags used, to which we add tags to discriminate among punctuation types. The original POS tag for punctuation was PU; we created new POS tags for each distinct punctuation type (e.g., PU-?).

The CTB corpus was collected during different time periods from different sources with a diversity of articles. In order to obtain a representative split of training, development, and test sets, we divide the whole corpus into blocks of 10 files by sorted order. For each block, the first file is used for development, the second file is used for test, and the remaining 8 files are used for training. Table 1 gives the basic statistics on the data. The development set is used to determine the parameter λ in Equation 4, the smoothing parameter ϵ in Equation 7, the weight parameter β described in Section 3.3, and the number of boosting rounds in the reranking model. In order to train the reranking model, the method in (Collins and Koo, 2005) is used to prepare the N-best training examples. We divided the training set into 20 chunks, with each chunk N-best tagged by the HMM model trained on the combination of the other 19 chunks. The development set is N-best tagged by the HMM model trained on the training set, and the test set is N-best tagged by the HMM model trained on the combination of the training set and the development set.

	Train	Dev	Test
#Sentences	14925	1904	1975
#Words	404844	51243	52900

Table 1: The basic statistics on the data.

In the following subsections, we will first examine the HMM models alone to determine the best HMM configuration to use to generate the N-best candidates, and then evaluate the reranking models. Finally, we compare our performance with previous work. In this paper, we use the sign test with $p \leq 0.01$ to evaluate the statistical significance of the difference between the performances of two models.

4.2 Results of the HMM taggers

The baseline HMM model ported directly from the English tagger, as described in Subsection 2.1, has an overall tag accuracy of 93.12% on the test set, which is fairly low compared to the 97% accuracy of many state-of-the-art taggers on WSJ for English.

By approximating the unknown word emission probability using the characters in the word as in Equation 3, the performance of the HMM tagger improves significantly to 93.43%, suggesting that characters in different positions of a Chinese word help to disambiguate the word class of the entire word, in contrast to English for which suffixes are most helpful.

Figure 1 depicts the impact of combining the left-to-right and right-to-left word emission models using different weighting values (i.e., λ) on the development set. Note that emission probabilities of unknown words are estimated based on characters using the same λ for combination. When $\lambda = 1.0$, the model uses only the standard left-to-right prediction of words, while when $\lambda = 0$ it uses only the right-to-left estimation. It is interesting to note that the right-to-left estimation results in greater accuracy than the left-to-right estimation. This might be because there is stronger interaction between a word and its next tag. Also as shown in Figure 1, the estimations in the two directions are complementary to each other, with $\lambda = 0.5$ performing best. The performance of the HMM taggers on the test set is given in Table 2 for the best operating point, as well as the two other extreme operating points to compare the left-to-right and right-to-left constraints. Our best HMM tagger further improves the tag accuracy significantly from 93.43% ($\lambda = 1.0$) to 94.01% ($\lambda = 0.5$).

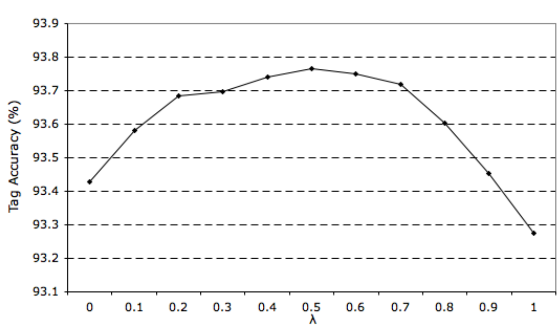

Figure 1: The accuracy of the HMM tagger on the development set with various λ values for combining the word emission probabilities.

	Overall	Known	Unknown
HMM baseline	93.12%	94.65%	69.08%
HMM, λ=1.0	93.43%	94.71%	73.41%
HMM, λ=0.0	93.65%	94.88%	74.23%
HMM, λ=0.5	94.01%	95.21%	75.15%

Table 2: The performance of various HMM taggers on the test set.

4.3 Results of the Reranking Models

The HMM tagger with the best accuracy (i.e., the one with $\lambda = 0.5$ in Table 2) is used to generate the N-Best tagging candidates, with a maximum of 100 candidates. As shown in Table 3, a maximum of 100-Best provides a reasonable margin for improvement in the reranking task.

We first test the performance of the reranking methods using only the n-gram feature set, which contains around 18 million features. Later, we will investigate the addition of morphological features and dependency features. The smoothing parameter ϵ (for Collins' method and the update-once method) and the weight parameter β (for the regularization method) both have great impact on reranking performance. We trained various reranking models with ϵ values of $0.0001 \times \{1, 2.5, 5, 7.5, 10, 25, 50, 75, 100\}$, and β values of $\{0.1, 0.25, 0.5, 0.75, 1\}$. For all these parameter values, 600,000 rounds of iterations were executed on the training set. The development set was used to determine the early stopping point in training. If not mentioned explicitly, all the results reported are based on the best parameters tuned on the development set.

	1-Best	50-Best	100-Best
train	93.48%	96.96%	97.13%
dev	93.75%	97.68%	97.84%
test	93.19%	97.19%	97.35%

Table 3: The oracle tag accuracies of the 1-Best, 50-Best, and 100-Best candidates in the training, development, and test sets for the reranking experiments. Note that the tagging candidates are prepared using the method described in Subsection 4.1.

Table 4 reports the performance of the best HMM tagger and the three reranking taggers on the test set. All three reranking methods improve the HMM tagger significantly. Also, the update-once and regularization methods both outperform Collins' original training method significantly.

	Overall	Known	Unknown
HMM, λ=0.5	94.01%	95.21%	75.15%
Collins	94.38%	95.56%	75.85%
Update-once	94.50%	95.67%	76.13%
Regularized	94.54%	95.70%	76.48%

Table 4: The performance on the test set of the HMM tagger, and the reranking methods using the n-gram features.

	Overall	Known	Unknown
HMM, λ=0.5	94.01%	95.21%	75.15%
Collins	94.44%	95.55%	77.05%
Update-once	94.68%	95.68%	78.91%
Regularized	94.64%	95.71%	77.84%

Table 6: The performance on the test set of the HMM tagger and the reranking methods using n-gram and morphological features.

We observed that no matter which value the smoothing parameter ϵ takes, there are only about 10,000 non-zero features finally selected by Collins' original method. In contrast, the two new methods select substantially more features, as shown in Table 5. As mentioned before, there are some strong features that only appear in positive or negative samples, i.e., either W_k^+ or W_k^- equals zero. Although introducing the smoothing parameter ϵ in Equation 7 prevents infinite weight values, the update to the feature weights is no longer optimal (in terms of minimizing the error function). Since the update is not optimal, subsequent iterations may still focus on these features (and thus ignore other weaker but informative features) and always increase their weights in one direction, leading to biased training.

The update-once method at each iteration selects a new feature that has the most impact in reducing the training loss function. It has the advantage of preventing increasingly large weights from being assigned to the strong features, enabling the update of other features. The regularization method allows multiple updates and also penalizes large weights. Once a feature is selected and has its weight updated, no matter how strong the feature is, the weight value is optimal in terms of the current weights of other features, so that the training algorithm would choose another feature to update. A previously selected feature may be selected again if it becomes suboptimal due to a change in the weights of other features.

	#iterations	#features	percent
Collins	115400	10020	8.68%
Update-once	545100	545100	100%
Regularized	92500	70131	75.82%

Table 5: The number of iterations (for the best performance), the number of selected features, and the percentage of selected features, by Collins' method, the update-once method, and the regularization method on the development set.

We next add morphological features to the n-gram features selected by the reranking methods[7]. As can be seen by comparing Table 6 to Table 4, morphological features improve the tagging accuracy of unknown words. It should be noted that the improvement made by both update-one and regularization methods is statistically significant over using n-gram features alone; however, the improvement by Collins' original method is not significant. This suggests that the two new methods are able to utilize a greater variety of features than the original method.

We trained several Charniak parsers using the same method for the HMM taggers to generate automatic parse trees for training, development, and test data. The update-once method is used to evaluate the effectiveness of dependency features for reranking, as shown in Table 7. The parser has an overall tagging accuracy that is greater than that of the best HMM tagger, but worse than that of the reranking models using n-gram and morphological features. It is interesting to note that reranking with the dependency features alone improves the tagging accuracy significantly, outperforming reranking models using n-gram and morphological features. This suggests that the long distance features based on the syntactic structure of the sentence are very beneficial for POS tagging of Mandarin. Moreover, n-gram and morphological features are complementary to the dependency features, with their combination performing the best. The n-gram features improve the accuracy on known words, while the morphological features improve the accuracy on unknown words. The best accuracy of 95.11% is an 18% relative reduction in error compared to the best HMM tagger.

[7]Because the size of the combined feature set of all n-gram features and morphological features is too large to be handled by our server, we chose to add morphological features to the n-gram features selected by the reranking methods, and then retrain the reranking model.

	Overall	Known	Unknown
Parser	94.31%	95.57%	74.52%
dep	94.93%	96.01%	77.87%
dep+ngram	95.00%	96.11%	77.49%
dep+morph	94.98%	96.01%	78.79%
dep+ngram+morph	95.11%	96.12%	79.32%

Table 7: The tagging performance of the parser and the update-once reranking models with dependency features and their combination with n-gram and morphological features.

4.4 Comparison to Previous Work

So how is our performance compared to previous work? When working on the same training/test data (CTB5.0 with the same pre-processing procedures) as in (Tseng et al., 2005), our HMM model obtained an accuracy of 93.72%, as compared to their 93.74% accuracy. Our reranking model[8] using n-gram and morphological features improves the accuracy to 94.16%. Note that we did not use all the morphological features as in (Tseng et al., 2005), which would probably provide additional improvement. The dependency features are expected to further improve the performance, although they are not included here in order to provide a relatively fair comparison.

5 Conclusions and Future Work

We have shown that the characters in a word are informative of the POS type of the entire word in Mandarin, reflecting the fact that the individual Chinese characters carry POS information to some degree. The syntactic relationship among characters may provide further information, which we leave as future work. We have also shown that the additional right-to-left estimation of word emission probabilities is useful for HMM tagging of Mandarin. This suggests that explicit modeling of bi-directional interactions captures more sequential information. This could possibly help in other sequential modeling tasks.

We have also investigated using the reranking algorithm in (Collins and Koo, 2005) for the Mandarin POS tagging task, and found it quite effective in improving tagging accuracy. The original algorithm has a tendency to focus on a small subset of strong features and ignore some of the other useful features. We were able to improve the performance of the reranking algorithm by utilizing two different methods that make better use of more features. Both are simple and yet effective. The effectiveness of dependency features suggests that syntax-based long distance features are important for improving part-of-speech tagging performance in Mandarin. Although parsing is computationally more demanding than tagging, we hope to identify related features that can be extracted more efficiently.

In future efforts, we plan to extract additional reranking features utilizing more explicitly the characteristics of Mandarin. We also plan to extend our work to speech transcripts for Broadcast News and Broadcast Conversation corpora, and explore semi-supervised training methods for reranking.

[8] Tseng at el.'s training/test split uses up the entire CTB corpus, leaving no development data for tuning parameters. In order to roughly measure reranking performance, we use the update-once method to train the reranking model for 600,000 rounds with the other parameters tuned in Section 4. This sacrifices performance to some extent.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. We gratefully acknowledge the comments from the anonymous reviewers.

References

Thorsten Brants. 2000. TnT a statistical part-of-speech tagger. In *ANLP*, pages 224–231.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 132–139, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

John Chen, Srinivas Bangalore, Michael Collins, and Owen Rambow. 2002. Reranking an n-gram supertagger. In *the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks*.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Michael Collins, Robert E. Schapire, and Yoram Singer. 2002. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1):253–285.

Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1(55):119–139.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. 1998. An efficient boosting algorithm for combining preferences. In *the Fifteenth International Conference on Machine Learning*.

Heng Ji, Cynthia Rudin, and Ralph Grishman. 2006. Reranking algorithms for name tagging. In *HLT/NAACL 06 Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*.

Fernando Pereira John Lafferty, Andrew McCallum. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Jerome Packard. 2000. *The Morphology of Chinese*. Cambridge University Press.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*.

Brian Roark, Yang Liu, Mary Harper, Robin Stewart, Matthew Lease, Matthew Snover, Izhak Shafran, Bonnie Dorr, John Hale, Anna Krasnyanskaya, and Lisa Yung. 2006. Reranking for sentence boundary detection in conversational speech. In *ICASSP*.

Scott M. Thede and Mary P. Harper. 1999. A second-order hidden Markov model for part-of-speech tagging. In *ACL*, pages 175–182.

Huihsin Tseng, Daniel Jurafsky, and Christopher Manning. 2005. Morphological features help pos tagging of unknown words across language varieties. In *the Fourth SIGHAN Workshop on Chinese Language Processing*.

Nianwen Xue, Fu dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *COLING*.