Improving A Simple Bigram HMM Part-of-Speech Tagger by Latent Annotation and Self-Training

Zhongqiang Huang[†], **Vladimir Eidelman**[†], **Mary Harper**[†‡]
[†]Laboratory for Computational Linguistics and Information Processing
Institute for Advanced Computer Studies
University of Maryland, College Park
[‡]Human Language Technology Center of Excellence
Johns Hopkins University
{zqhuang,vlad,mharper}@umiacs.umd.edu

Abstract

In this paper, we describe and evaluate a bigram part-of-speech (POS) tagger that uses latent annotations and then investigate using additional genre-matched unlabeled data for self-training the tagger. The use of latent annotations substantially improves the performance of a baseline HMM bigram tagger, outperforming a trigram HMM tagger with sophisticated smoothing. The performance of the latent tagger is further enhanced by self-training with a large set of unlabeled data, even in situations where standard bigram or trigram taggers do not benefit from self-training when trained on greater amounts of labeled training data. Our best model obtains a state-of-the-art Chinese tagging accuracy of 94.78% when evaluated on a representative test set of the Penn Chinese Treebank 6.0.

1 Introduction

Part-of-speech (POS) tagging, the process of assigning every word in a sentence with a POS tag (e.g., NN (normal noun) or JJ (adjective)), is prerequisite for many advanced natural language processing tasks. Building upon the large body of research to improve tagging performance for various languages using various models (e.g., (Thede and Harper, 1999; Brants, 2000; Tseng et al., 2005b; Huang et al., 2007)) and the recent work on PCFG grammars with latent annotations (Matsuzaki et al., 2005; Petrov et al., 2006), we will investigate the use of fine-grained latent annotations for Chinese POS tagging. While state-of-the-art tagging systems have achieved accuracies above 97% in English, Chinese POS tagging (Tseng et al., 2005b; Huang et al., 2007) has proven to be more challenging, and it is the focus of this study.

The value of the latent variable approach for tagging is that it can learn more fine grained tags to better model the training data. Liang and Klein (2008) analyzed the errors of unsupervised learning using EM and found that both estimation and optimization errors decrease as the amount of unlabeled data increases. In our case, the learning of latent annotations through EM may also benefit from a large set of automatically labeled data to improve tagging performance. Semi-supervised, self-labeled data has been effectively used to train acoustic models for speech recognition (Ma and Schwartz, 2008); however, early investigations of self-training on POS tagging have mixed outcomes. Clark et al. (2003) reported positive results with little labeled training data but negative results when the amount of labeled training data increases. Wang et al. (2007) reported that self-training improves a trigram tagger's accuracy, but this tagger was trained with only a small amount of in-domain labeled data.

In this paper, we will investigate whether the performance of a simple bigram HMM tagger can be improved by introducing latent annotations and whether self-training can further improve its performance. To the best of our knowledge, this is the first attempt to use latent annotations with self-training to enhance the performance of a POS tagger.

2 Model

POS tagging using a hidden Markov model can be considered as an instance of Bayesian inference,

Proceedings of NAACL HLT 2009: Short Papers, pages 213–216,
Boulder, Colorado, June 2009.

wherein we observe a sequence of words and need to assign them the most likely sequence of POS tags. If t_1^i denotes the tag sequence $t_1, \cdots, t_i$, and w_1^i denotes the word sequence $w_1, \cdots, w_i$, given the first-order Markov assumption of a bigram tagger, the best tag sequence $\tau(w_1^n)$ for sentence w_1^n can be computed efficiently as[1]:

$$\begin{aligned}\tau(w_1^n) &= \arg\max_{t_1^n} p(t_1^n|w_1^n) \\ &\approx \arg\max_{t_1^n} \prod_i p(t_i|t_{i-1})p(w_i|t_i)\end{aligned}$$

with a set of transition parameters $\{p(b|a)\}$, for transiting to tag b from tag a, and a set of emission parameters $\{p(w|a)\}$, for generating word w from tag a. A simple HMM tagger is trained by pulling counts from labeled data and normalizing to get the conditional probabilities.

It is well know that the independence assumption of a bigram tagger is too strong in many cases. A common practice for weakening the independence assumption is to use a second-order Markov assumption, i.e., a trigram tagger. This is similar to explicitly annotating each POS tag with the preceding tag. Rather than explicit annotation, we could use latent annotations to split the POS tags, similarly to the introduction of latent annotations to PCFG grammars (Matsuzaki et al., 2005; Petrov et al., 2006). For example, the NR tag may be split into NR-1 and NR-2, and correspondingly the POS tag sequence of "Mr./NR Smith/NR saw/VV Ms./NR Smith/NR" could be refined as: "Mr./NR-2 Smith/NR-1 saw/VV-2 Ms./NR-2 Smith/NR-1".

The objective of training a bigram tagger with latent annotations is to find the transition and emission probabilities associated with the latent tags such that the likelihood of the training data is maximized. Unlike training a standard bigram tagger where the POS tags are observed, in the latent case, the latent tags are not observable, and so a variant of EM algorithm is used to estimate the parameters.

Given a sentence w_1^n and its tag sequence t_1^n, consider the i-th word w_i and its latent tag $a_x \in a = t_i$ (which means a_x is a latent tag of tag a, the i-th tag in the sequence) and the $(i+1)$-th word w_{i+1} and its latent tag $b_y \in b = t_{i+1}$, the forward, $\alpha_{i+1}(b_y) = p(w_1^{i+1}, b_y)$, and backward, $\beta_i(a_x) = p(w_{i+1}^n|a_x)$, probabilities can be computed recursively:

$$\alpha_{i+1}(b_y) = \sum_x \alpha_i(a_x)p(b_y|a_x)p(w_{i+1}|b_y)$$

$$\beta_i(a_x) = \sum_y p(b_y|a_x)p(w_{i+1}|b_y)\beta_{j+1}(b_y)$$

In the E step, the posterior probabilities of co-occurrence events can be computed as:

$$\begin{aligned}p(a_x, b_y|w) &\propto \alpha_i(a_x)p(b_y|a_x)\beta_{i+1}(b_y) \\ p(a_x, w_i|w) &\propto \alpha_i(a_x)\beta_i(a_x)\end{aligned}$$

In the M step, the above posterior probabilities are used as weighted observations to update the transition and emission probabilities[2]:

$$\begin{aligned}p(b_y|a_x) &= c(a_x, b_y)/\sum_{b_y} c(a_x, b_y) \\ p(w|a_x) &= c(a_x, w)/\sum_w c(a_x, w)\end{aligned}$$

A hierarchical split-and-merge method, similar to (Petrov et al., 2006), is used to gradually increase the number of latent annotations while allocating them adaptively to places where they would produce the greatest increase in training likelihood (e.g., we observe heavy splitting in categories such as NN (normal noun) and VV (verb), that cover a wide variety of words, but only minimal splitting in categories like IJ (interjection) and ON (onomatopoeia)).

Whereas tag transition occurrences are frequent, allowing extensive optimization using EM, word-tag co-occurrences are sparser and more likely to suffer from over-fitting. To handle this problem, we map all words with frequency less than threshold[3] λ to symbol *unk* and for each latent tag accumulate the word tag statistics of these rare words to $c_r(a_x, \textit{unk}) = \sum_{w:c(w)<\lambda} c(a_x, w)$. These statistics are redistributed among the rare words $(w : c(w) < \lambda)$ to compute their emission probabilities:

$$\begin{aligned}c(a_x, w) &= c_r(a_x, \textit{unk}) \cdot c(a, w)/c_r(a, unk) \\ p(w|a_x) &= c(a_x, w)/\sum_w c(a_x, w)\end{aligned}$$

The impact of this rare word handling method will be investigated in Section 3.

A character-based unknown word model, similar to the one described in (Huang et al., 2007), is used to handle unknown Chinese words during tagging. A decoding method similar to the max-rule-product method in (Petrov and Klein, 2007) is used to tag sentences using our model.

3 Experiments

The Penn Chinese Treebank 6.0 (CTB6) (Xue et al., 2005) is used as the labeled data in our study. CTB6

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

[2] $c(\cdot)$ represents the count of the event.

[3] The value of λ is tuned on the development set.

contains news articles, which are used as the primary source of labeled data in our experiments, as well as broadcast news transcriptions. Since the news articles were collected during different time periods from different sources with a diversity of topics, in order to obtain a representative split of train-test-development sets, we divide them into blocks of 10 files in sorted order and for each block use the first file for development, the second for test, and the remaining for training. The broadcast news data exhibits many of the characteristics of newswire text (it contains many nonverbal expressions, e.g., numbers and symbols, and is fully punctuated) and so is also included in the training data set. We also utilize a greater number of unlabeled sentences in the self-training experiments. They are selected from similar sources to the newswire articles, and are normalized (Zhang and Kahn, 2008) and word segmented (Tseng et al., 2005a). See Table 1 for a summary of the data used.

	Train	Dev	Test	Unlabeled
sentences	24,416	1904	1975	210,000
words	678,811	51,229	52,861	6,254,947

Table 1: The number of sentences and words in the data.

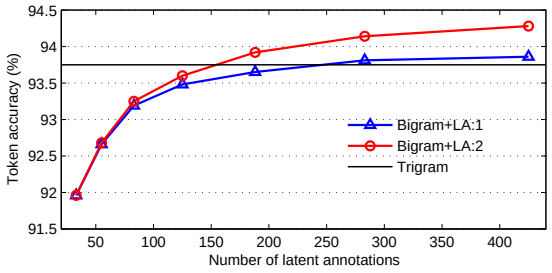

Figure 1: The learning curves of the bigram tagger with latent annotations on the development set.

Figure 1 plots the learning curves of two bigram taggers with latent annotations (Bigram+LA:2 has the special handling of rare words as described in Section 2 while Bigram+LA:1 does not) and compares its performance with a state-of-the-art trigram HMM tagger (Huang et al., 2007) that uses trigram transition and emission models together with bidirectional decoding. Both bigram taggers initially have much lower tagging accuracy than the trigram tagger, due to its strong but invalid independence assumption. As the number of latent annotations increases, the bigram taggers are able to learn more from the context based on the latent annotations, and their performance improves significantly, outperforming the trigram tagger. The performance gap between the two bigram taggers suggests that over-fitting occurs in the word emission model when more latent annotations are available for optimization; sharing the statistics among rare words alleviates some of the sparseness while supporting the modeling of deeper dependencies among more frequent events. In the later experiments, we use Bigram+LA to denote the Bigram+LA:2 tagger.

Figure 2 compares the self-training capability of three models (the bigram tagger w/ or w/o latent annotations, and the aforementioned trigram tagger) using different sizes of labeled training data and the full set of unlabeled data. For each model, a tagger is first trained on the allocated labeled training data and is then used to tag the unlabeled data. A new tagger is then trained on the combination[4] of the allocated labeled training data and the newly automatically labeled data.

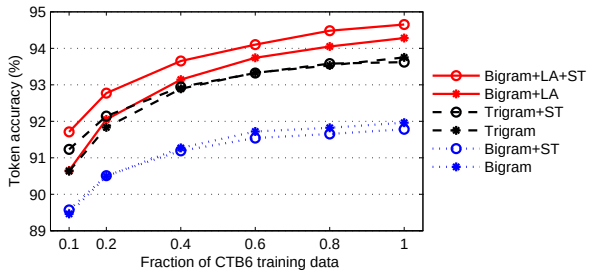

Figure 2: The performance of three taggers evaluated on the development set, before and after self-training with different sizes of labeled training data.

There are two interesting observations that distinguish the bigram tagger with latent annotations from the other two taggers. First, although all of the taggers improve as more labeled training data is available, the performance gap between the bigram tagger with latent annotations and the other two taggers also increases. This is because more latent annotations can be used to take advantage of the additional training data to learn deeper dependencies.

Second, the bigram tagger with latent annotations benefits much more from self-training, although it

[4]We always balance the size of manually and automatically labeled data through duplication (for the trigram tagger) or posterior weighting (for the bigram tagger w/ or w/o latent annotations), as this provides superior performance.

already has the highest performance among the three taggers before self-training. The bigram tagger without latent annotations benefits little from self-training. Except for a slight improvement when there is a small amount of labeled training, self-training slightly hurts tagging performance as the amount of labeled data increases. The trigram tagger benefits from self-training initially but eventually has a similar pattern to the bigram tagger when trained on the full labeled set. The performance of the latent bigram tagger improves consistently with self-training. Although the gain decreases for models trained on larger training sets, since stronger models are harder to improve, self-training still contributes significantly to model accuracy.

The final tagging performance on the test set is reported in Table 2. All of the improvements are statistically significant ($p < 0.005$).

Tagger	Token Accuracy (%)
Bigram	92.25
Trigram	93.99
Bigram+LA	94.53
Bigram+LA+ST	94.78

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

It is worth mentioning that we initially added latent annotations to a trigram tagger, rather than a bigram tagger, to build from a stronger starting point; however, this did not work well. A trigram tagger requires sophisticated smoothing to handle data sparsity, and introducing latent annotations exacerbates the sparsity problem, especially for trigram word emissions. The uniform extension of a bigram tagger to a trigram tagger ignores whether the use of additional context is helpful and supported by enough data, nor is it able to use a longer context. In contrast, the bigram tagger with latent annotations is able to learn different granularities for tags based on the training data.

4 Conclusion

In this paper, we showed that the accuracy of a simple bigram HMM tagger can be substantially improved by introducing latent annotations together with proper handling of rare words. We also showed that this tagger is able to benefit from self-training, despite the fact that other models, such as bigram or trigram HMM taggers, do not.

In the future work, we will investigate automatic data selection methods to choose materials that are most suitable for self-training and evaluate the effect of the amount of automatically labeled data.

Acknowledgments

This work was supported by NSF IIS-0703859 and DARPA HR0011-06-C-0023 and HR0011-06-2-001. Any opinions, findings and/or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

T. Brants. 2000. TnT a statistical part-of-speech tagger. In *ANLP*.

S. Clark, J. R. Curran, and M. Osborne. 2003. Bootstrapping pos taggers using unlabelled data. In *CoNLL*.

Z. Huang, M. Harper, and W. Wang. 2007. Mandarin part-of-speech tagging and discriminative reranking. *EMNLP*.

P. Liang and D. Klein. 2008. Analyzing the errors of unsupervised learning. In *ACL*.

J. Ma and R. Schwartz. 2008. Factors that affect unsupervised training of acoustic models. In *Interspeech*.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*. Association for Computational Linguistics.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.

S. M. Thede and M. P. Harper. 1999. A second-order hidden markov model for part-of-speech tagging. In *ACL*.

H. Tseng, P. Chang, G. Andrew, D. Jurafsky, and C. Manning. 2005a. A conditional random field word segmenter. In *SIGHAN Workshop on Chinese Language Processing*.

H. Tseng, D. Jurafsky, and C. Manning. 2005b. Morphological features help pos tagging of unknown words across language varieties. In *SIGHAN Workshop on Chinese Language Processing*.

W. Wang, Z. Huang, and M. Harper. 2007. Semi-supervised learning for part-of-speech tagging of Mandarin transcribed speech. In *ICASSP*.

N. Xue, F. Xia, F. Chiou, and M. Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*.

B. Zhang and J. G. Kahn. 2008. Evaluation of decatur text normalizer for language model training. Technical report, University of Washington.