Training a Perceptron with Global and Local Features for Chinese Word Segmentation

Dong Song and **Anoop Sarkar**
School of Computing Science, Simon Fraser University
Burnaby, BC, Canada V5A1S6
{dsong,anoop}@cs.sfu.ca

Abstract

This paper proposes the use of global features for Chinese word segmentation. These global features are combined with local features using the averaged perceptron algorithm over N-best candidate word segmentations. The N-best candidates are produced using a conditional random field (CRF) character-based tagger for word segmentation. Our experiments show that by adding global features, performance is significantly improved compared to the character-based CRF tagger. Performance is also improved compared to using only local features. Our system obtains an F-score of 0.9355 on the CityU corpus, 0.9263 on the CKIP corpus, 0.9512 on the SXU corpus, 0.9296 on the NCC corpus and 0.9501 on the CTB corpus. All results are for the closed track in the fourth SIGHAN Chinese Word Segmentation Bakeoff.

1 Introduction

Most natural language processing tasks require that the input be tokenized into individual words. For some languages, including Chinese, this is challenging since the sentence is typically written as a string of characters without spaces between words. Word segmentation is the task of recovering the most plausible grouping of characters into words. In this paper, we describe the system we developed for the fourth SIGHAN Chinese Word Segmentation Bakeoff[1]. We test our system in the closed track[2] for all five corpora: Academia Sinica (CKIP), City University of Hong Kong (CityU), National Chinese Corpus (NCC), University of Colorado (CTB), and Shanxi University (SXU).

2 System Description

The architecture of our system is shown in Figure 1. For each of the training corpora in the bakeoff, we produce a 10-fold split: in each fold, 90% of the corpus is used for training and 10% is used to produce an N-best list of candidates. The N-best list is produced using a character-based conditional random field (CRF) (Lafferty et al., 2001; Kudo et al., 2004) tagger. The true segmentation can now be compared with the N-best list in order to train an averaged perceptron algorithm (Collins, 2002a). This system is then used to predict the best word segmentation from an N-best list for each sentence in the test data.

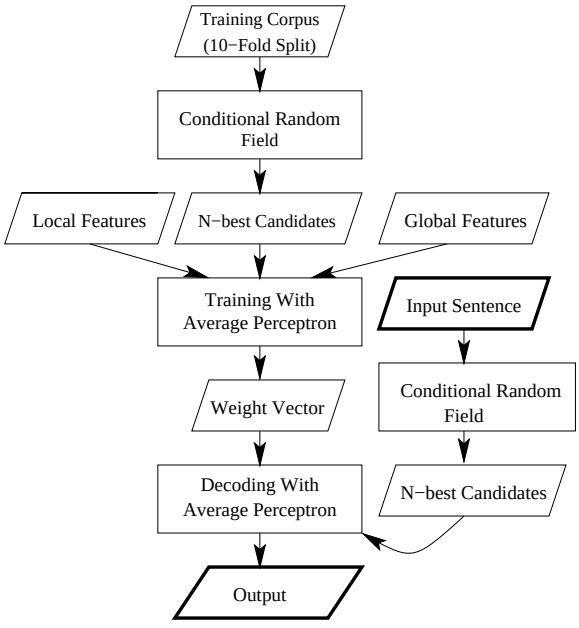

Figure 1: Outline of the segmentation process

2.1 Learning Algorithm

Given an unsegmented sentence x, the word segmentation problem can be defined as finding the

[1] Further details at: www.china-language.gov.cn/bakeoff08/bakeoff-08_basic.html

[2] We do not use any extra annotation, especially for punctuation, dates, numbers or English letters.

most probable segmentation $F(x)$ from a set of possible segmentations of x.

$$F(x) = \underset{y \in GEN(x)}{\operatorname{argmax}} \Phi(x, y) \cdot \overline{w} \quad (1)$$

The set of possible segmentations is given by *GEN*(x) and the naïve method is to first generate all possible segmented candidates. For a long sentence, generating those candidates and picking the one with the highest score is time consuming.

In our approach, N-best candidates for each training example are produced with the *CRF++* software (Kudo et al., 2004). The CRF is used as a tagger that tags each character with the following tags: for each multi-character word, its first character is given a **B** (Beginning) tag , its last character is assigned an **E** (End) tag, while each of its remaining characters is provided an **M** (Middle) tag. In addition, for a single-character word, **S** (Single) is used as its tag[3]. Let c_0 be the current character, c_{-1}, c_{-2} are the two preceding characters, and c_1, c_2 are the two characters to the right . Using this notation, the features used in our CRF models are: c_0, c_{-1}, c_1, c_{-2}, c_2, $c_{-1}c_0$, c_0c_1, $c_{-1}c_1$, $c_{-2}c_{-1}$ and c_0c_2.

We use the now standard method for producing N-best candidates in order to train our re-ranker which uses global and local features: 10-folds of training data are used to train the tagger on 90% of the data and then produce N-best lists for the remaining 10%. This process gives us an N-best candidate list for each sentence and the candidate that is most similar to the true segmentation, called y^b. We map a segmentation y to *features* associated with the segmentation using the mapping $\Phi(\cdot)$. The score of a segmentation y is provided by the dot-product $\Phi(y) \cdot \overline{w}$. The perceptron algorithm (Fig. 2) finds the weight parameter vector $\overline{w}$ using online updates. The predicted segmentation $y_i^{'}$ based on the current weight vector is compared to the the best candidate y^b, and whenever there is a mismatch, the algorithm updates the parameter vector by incrementing the parameter value for features in y^b, and by decrementing the value for features in $y_i^{'}$.

The voted perceptron (Freund and Schapire, 1999) has considerable advantages over the standard

[3]Note that performance of the CRF tagger could be improved with the use of other tagsets. However, this does not affect our comparative experiments in this paper.

Inputs: Training Data $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$
Initialization: Set $\overline{w} = 0$
Algorithm:

```
for t = 1, ..., T do
  for i = 1, ..., m do
    Calculate y_i', where
    y_i' = argmax_{y ∈ N-best Candidates} Φ(y) · w̄
    if y_i' ≠ y^b then
      w̄ = w̄ + Φ(y^b) − Φ(y_i')
    end if
  end for
end for
```

Figure 2: Training using a perceptron algorithm over N-best candidates.

perceptron. However, due to the computational issues with the voted perceptron, the averaged perceptron algorithm (Collins, 2002a) is used instead. Rather than using $\overline{w}$, we use the averaged weight parameter $\overline{\gamma}$ over the m training examples for future predictions on unseen data:

$$\overline{\gamma} = \frac{1}{mT} \sum_{i=1..m, t=1..T} \overline{w}^{i,t}$$

In calculating $\overline{\gamma}$, an accumulating parameter vector $\overline{\sigma}^{i,t}$ is maintained and updated using $\overline{w}$ for each training example; therefore, $\overline{\sigma}^{i,t} = \sum \overline{w}^{i,t}$. After the last iteration, $\overline{\sigma}^{i,t}/mT$ produces the final parameter vector $\overline{\gamma}$.

When the number of features is large, it is time consuming to calculate the total parameter $\overline{\sigma}^{i,t}$ for each training example. To reduce the time complexity, we adapted the lazy update proposed in (Collins, 2002b), which was also used in (Zhang and Clark, 2007). After processing each training sentence, not all dimensions of $\overline{\sigma}^{i,t}$ are updated. Instead, an update vector $\overline{\tau}$ is used to store the exact location (i, t) where each dimension of the averaged parameter vector was last updated, and only those dimensions corresponding to features appearing in the current sentence are updated. While for the last example in the last iteration, each dimension of $\overline{\tau}$ is updated, no matter whether the candidate output is correct.

2.2 Feature Templates

The feature templates used in our system include both local features and global features. For local fea-

tures, we consider two major categories: word-based features and character-based features. Five specific types of features from (Zhang and Clark, 2007) that are shown in Table 1 were used in our system. In our initial experiments, the other features used in (Zhang and Clark, 2007) did not improve performance and so we do not include them in our system.

1	word w
2	word bigram w_1w_2
3	single character word w
4	space-separated characters c_1 and c_2
5	character bi-gram c_1c_2 in any word

Table 1: local feature templates. Rows 1, 2 and 3 are word-based and rows 4 and 5 are character-based features

In our system, we also used two types of global features per sentence (see Table 2). By global, we mean features over the entire segmented sentence.[4]

6	sentence confidence score
7	sentence language model score

Table 2: global feature template

The sentence confidence score is calculated by *CRF++* during the production of the N-best candidate list, and it measures how confident each candidate is close to the true segmentation.

The sentence language model score for each segmentation candidate is produced using the SRILM toolkit (Stolcke, 2002) normalized using the formula $P^{1/L}$, where P is the probability-based language model score and L is the length of the sentence in words (not in characters). For global features, the feature weights are not learned using the perceptron algorithm but are determined using a development set.

3 Experiments and Analysis

Our system is tested on all five corpora provided in the fourth SIGHAN Bakeoff, in the closed track.

3.1 Parameter Pruning

First, the value of the parameter N, which is the maximum number of N-best candidates, was determined. An oracle procedure proceeds as follows: 80% of the training corpus is used to train the CRF model, and produce N-best outputs for each sentence on the remaining 20% of the data. Then these N candidates are compared with the true segmentation, and for each training sentence, the candidate closest to the truth is chosen as the final output. Testing on different values of N, we chose N to be 20 in all our experiments since that provided the best tradeoff between accuracy and speed.

Next, the weight for sentence confidence score S_{crf} and that for language model score S_{lm} are determined. To simplify the process, we assume that the weights for both S_{crf} and S_{lm} are equal. In this step, each training corpus is separated into a training set (80% of the whole corpus) and a held-out set (20% of the corpus). Then, the perceptron algorithm is applied on the training set with different S_{crf} and S_{lm} values, and for various number of iterations. The weight values we test include 2, 4, 6, 8, 10, 20, 30, 40, 50, 100 and 200. From the experiments, the weights are chosen to be 100 for CKIP corpus, 10 for CityU corpus, 30 for NCC corpus, 20 for CTB corpus, and 10 for SXU corpus.

While determining the weights for global features, the number of training iterations can be determined as well. Experiments show that, as the number of iterations increases, the accuracy stabilizes in most cases, reflecting the convergence of the learning algorithm. Analyzing the learning curves, we fix the number of training iterations to be 5 for CKIP corpus, 9 for NCC corpus, and 8 for the CityU, CTB and SXU corpora.

3.2 Results on the Fourth SIGHAN Bakeoff

In each experiment, F-score (F) is used to evaluate the segmentation accuracy. Table 3 shows the F-score on the fourth SIGHAN Bakeoff corpora. In this table, we record the performance of our system, the score from the character-based CRF method and the score from the averaged perceptron using only local features.

Our system outperforms the baseline character-based CRF tagger. In addition, the use of global features in the re-ranker produces better results than only using local features.

The only data set on which the performance of our system is lower than the character-based CRF method is CKIP corpus. For this data set during the

[4]It is important to distinguish this kind of global feature from another type of 'global' feature that either enforces consistency or examines the use of a feature in the entire training or testing corpus.

	CKIP	NCC	CityU	CTB	SXU
Character-based CRF method	0.9332	0.9248	0.9320	0.9468	0.9473
Averaged Perceptron with only local features	0.9180	0.9125	0.9273	0.9450	0.9387
Our System	0.9263	**0.9296**	**0.9355**	**0.9501**	**0.9512**
Our System (With modified weight for global features)	**0.9354**	–	–	–	–
Significance (p-value)	$\leq$ 1.19e-12	$\leq$ 4.43e-69	$\leq$ 3.55e-88	$\leq$ 2.17e-18	$\leq$ 2.18e-38

Table 3: F-scores on the Fourth SIGHAN Bakeoff Corpora

parameter pruning step, the weight for S_{crf} and S_{lm} was too large. By lowering the weight from 100 to 4, we obtains an F-score of 0.9354, which is significantly better than the baseline CRF tagger.

The significance values in Table 3 were produced using the McNemar's Test (Gillick, 1989)[5]. All our results are significantly better.

4 Related Work

Re-ranking over N-best lists has been applied to so many tasks in natural language that it is not possible to list them all here. Closest to our approach is the work in (Kazama and Torisawa, 2007). They proposed a margin perceptron approach for named entity recognition with non-local features on an N-best list. In contrast to their approach, in our system, global features examine the entire sentence instead of partial phrases. For word segmentation, (Wang and Shi, 2006) implemented a re-ranking method with POS tagging features. In their approach, character-based CRF model produces the N-best list for each test sentence. The Penn Chinese TreeBank is used to train a POS tagger, which is used in re-ranking. However the POS tags are used as local and not global features. Note that we would not use POS tags in the closed track.

5 Conclusion

We have participated in the closed track of the fourth SIGHAN Chinese word segmentation bakeoff, and we provide results on all five corpora. We have shown that by combining global and local features, we can improve accuracy over simply using local features, and we also show improved accuracy over the baseline CRF character-based tagger for word segmentation.

[5] www.fon.hum.uva.nl/Service/Statistics/McNemars_test.html

References

M. Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proc. of the Empirical Methods in Natural Language Processing (EMNLP). MACL, 2002*, 1–8, 2000.

M. Collins. 2002. Ranking Algorithms for Named-Entity Extractions: Boosting and the Voted Perceptron. In *Proc. of ACL 2002*.

Y. Freund and R. Schapire. 1999. Large Margin Classification using the Perceptron Algorithm. In *Machine Learning*, 37(3): 277–296.

L. Gillick and S. Cox. 1989. Some Statistical Issues in the Comparison of Speech Recognition Algorithms. In *Proc. of IEEE Conf. on Acoustics, Speech and Sig. Proc., Glasgow, 1989*, 532–535.

J. Kazama and K. Torisawa. 2007. A New Perceptron Algorithm for Sequence Labeling with Non-local Features. In *Proc. of EMNLP-CoNLL 2007* , pages 315–324.

T. Kudo, K. Yamamoto, and Y. Matsumoto. 2004. Appliying Conditional Random Fields to Japanese Morphological Analysis. In *Proc. of EMNLP 2004*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of ICML-2001*, pages 591–598.

A. Ratnaparkhi. 1996. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proc. of EMNLP 1996*.

A. Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proc. Intl. Conf. Spoken Language Processing, Denver, Colorado, September 2002*.

M. Wang and Y. Shi. 2006. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proc. of the Fifth SIGHAN Workshop on Chinese Language Processing, Sydney, 2006*, pages 205–208.

Y. Zhang and S. Clark. 2007. Chinese Segmentation with a Word-based Perceptron Algorithm. In *Proc. of ACL 2007*.