Combining Classifiers for Chinese Word Segmentation

Nianwen Xue

Institute for Research in Cognitive Science
University of Pennsylvania
Suite 400A, 3401 Walnut
Philadelphia, PA 19014
xueniwen@linc.cis.upenn.edu

Susan P. Converse

Dept. of Computer and Information Science
University of Pennsylvania
200 South 33rd Street,
Philadelphia, PA 19104-6389
spc@linc.cis.upenn.edu

Abstract

In this paper we report results of a supervised machine-learning approach to Chinese word segmentation. First, a maximum entropy tagger is trained on manually annotated data to automatically labels the characters with tags that indicate the position of character within a word. An error-driven transformation-based tagger is then trained to clean up the tagging inconsistencies of the first tagger. The tagged output is then converted into segmented text. The preliminary results show that this approach is competitive compared with other supervised machine-learning segmenters reported in previous studies.

1 Introduction

It is generally agreed among researchers that word segmentation is a necessary first step in Chinese language processing. Most of the previous work in this area views a good dictionary as the cornerstone of this task. Several word segmentation algorithms have been developed using a dictionary as an essential tool. Most notably, variants of the maximum matching algorithm have been applied to word segmentation with considerable success. The results that have been reported are generally in the upper 90 percentile range. However, the success of such algorithms is premised on a large, exhaustive dictionary. The accuracy of word segmentation degrades sharply as new words appear. Since Chinese word formation is a highly productive process, new words are bound to appear in substantial numbers in realistic scenarios (Wu and Jiang 1998, Xue 2001), and it is virtually impossible to list all the words in a dictionary. In recent years, as annotated Chinese corpora have become available, various machine-learning approaches have been applied to Chinese word segmentation, with different levels of success. Compared with dictionary-based approaches, machine-learning approaches have the advantage of not needing a dictionary and thus are more suitable for use on naturally occurring Chinese text. In this paper we report results of a supervised machine-learning approach towards Chinese word segmentation that combines two fairly standard machine learning models. We show that this approach is very promising compared with dictionary-based approaches as well as other machine-learning approaches that have been reported in the literature.

2 Combining Classifiers for Chinese word segmentation

The two machine-learning models we use in this work are the maximum entropy model (Ratnaparkhi 1996) and the error-driven transformation-based learning model (Brill 1994). We use the former as the main workhorse and the latter to correct some of the errors produced by the former.

2.1 Reformulating word segmentation as a tagging problem

Before we apply the machine-learning algorithms we first convert the manually segmented words in the corpus into a tagged

sequence of Chinese characters. To do this, we tag each character with one of the four tags, LL, RR, MM and LR, depending on its position within a word. It is tagged LL if it occurs on the left boundary of a word, and forms a word with the character(s) on its right. It is tagged RR if it occurs on the right boundary of a word, and forms a word with the character(s) on its left. It is tagged MM if it occurs in the middle of a word. It is tagged LR if it forms a word by itself. We call such tags position-of-character (POC) tags to differentiate them from the more familiar part-of-speech (POS) tags. For example, the manually segmented string in (1)a will be tagged as shown in (1)b:

(1) a. 上海 计划 到 本 世纪 末 实现 人均 国内 生产 总值 五千 美元

b. 上/LL 海/RR 计/LL 划/RR 到/LR 本/LR 世/LL 纪/RR 末/LR 实/LL 现/RR 人/LL 均/RR 国/LL 内/RR 生/LL 产/RR 总/LL 值/RR 五/LL 千/RR 美/LL 元/RR

c. Shanghai plans to reach the goal of 5,000 dollars in per capita GDP by the end of the century.

Given a manually segmented corpus, a POC-tagged corpus can be derived trivially with perfect accuracy. The reason that we use such POC-tagged sequences of characters instead of applying n-gram rules to a segmented corpus directly (Hockenmaier and Brew 1998, Xue 2001) is that they are much easier to manipulate in the training process. Naturally, while some characters will have only one POC tag, most characters will receive multiple POC tags, in the same way that words can have multiple POS tags. The example in (2) shows how all four of the POC tags can be assigned to the character 产 (‘produce’):

(2) 产 LL 产品 'product'
产 LR 产 'produce'
产 MM 生产力 'productivity'
产 RR 投产 'start production'

Also as in POS tags, the way the character is POC-tagged in naturally occurring text is affected by the context in which it occurs. For example, if the preceding character is tagged a LR or RR, then the next character can only be tagged LL or LR. How a character is tagged is also affected by the surrounding characters. For example, 关 (‘close’) should be tagged RR if the previous character is 开 (‘open’) and neither of them forms a word with other characters, while it should be tagged LL if the next character is 心 (‘heart’) and neither of them forms a word with other characters. This state of affairs closely resembles the familiar POS tagging problem and lends itself naturally to a solution similar to that of POS tagging. The task is one of ambiguity resolution in which the correct POC tag is determined among several possible POC tags in a specific context. Our next step is to train a maximum entropy model on the perfectly POC-tagged data derived from a manually segmented corpus and use the model to automatically POC-tag unseen text.

2.2 The maximum entropy tagger

The maximum entropy model used in POS-tagging is described in detail in Ratnaparkhi (1996) and the POC tagger here uses the same probability model. The probability model is defined over $H \times T$, where H is the set of possible contexts or "histories" and T is the set of possible tags. The model's joint probability of a history h and a tag t is defined as

$$p(h,t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j^{(h,t)}} \qquad \text{(i)}$$

where π is a normalization constant, $\{\mu, \alpha_1, ..., \alpha_k\}$ are the model parameters and $\{f_1, ..., f_k\}$ are known as features, where $f_j\ (h, t) \in \{0,1\}$. Each feature f_j has a corresponding parameter α_j, which effectively serves as a "weight" of this feature. In the training process, given a sequence n of characters $\{c_1,\ldots,c_n\}$ and their POC tags $\{t_1,...,t_n\}$ as training data, the purpose is to determine the parameters $\{\mu, \alpha_1, ..., \alpha_k\}$ that maximize the likelihood L of the training data using p:

$$L(p) = \prod_{i=1}^{n} p(h_i, t_i) = \prod_{i=1}^{n} \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j^{(h_i,t_i)}} \qquad \text{(ii)}$$

The success of the model in tagging depends to a large extent on the selection of suitable features. Given *(h,t)*, a feature must encode information that helps to predict *t*. The features we used in this experiment are instantiations of the following feature templates:

(3) Feature templates used in this tagger:

a. The current character

b.The previous (next) character and the current character

c. The previous (next) two characters

d. The tag of the previous character

e. The tag of the character two before the current character

f. Whether the current character is a punctuation mark

g. Whether the current character is a numeral

h. Whether the current character is a Latin letter

In general, given *(h,t)*, these features are in the form of co-occurrence relations between *t* and some type of context. For example,

$$f_i(h_i, t_i) = \begin{cases} 1 & if\ t_{i-1} = \quad LL \,\&\, t_i = RR \\ 0 & otherwise \end{cases}$$

This feature will map to 1 and contribute towards $p(h_i,t_i)$ if $c_{(i-1)}$ is tagged LL and c_i is tagged RR.

The feature templates in (3) encode three types of contexts. First, features based on the current and surrounding characters are extracted. Given a character in a sentence, this model will look at the current character, the previous two and next two characters. For example, if the current character is 化 (‘-ize’), it is very likely that it will occur as a suffix in a word, thus receiving the tag RR. On the other hand, other characters might be equally likely to appear on the left, on the right or in the middle. In those cases, where a character occurs within a word depends on its surrounding characters. For example, if the current character is 爱 (‘love’), it should perhaps be tagged LL if the next character is 护 (‘protect’). However, if the previous character is 热 (‘warm’), then it should perhaps be tagged RR.

In the second type of context, features based on the previous tags are extracted. Information like this is useful in predicting the POC tag for the current character just as the POS tags are useful in predicting the POS tag of the current word in a similar context. For example, if the previous character is tagged LR or RR, this means that the current character must start a word, and should be tagged either LL or LR. Finally, limited POS-tagging information can also be used to predict how the current character should be POC-tagged. For example, a punctuation mark is generally treated as one segment in the CTB corpus. Therefore, if a character is a punctuation mark, then it should be POC-tagged LR. This also means that the previous character should close a word and the following character should start a word. When the training is complete, the features and their corresponding parameters will be used to calculate the probability of the tag sequence of a sentence when the tagger tags unseen data. Given a sequence of characters $\{c_1,...,c_n\}$, the tagger searches for the tag sequence $\{t_1, ..., t_n\}$ with the highest conditional probability

$$P(t_1,...t_n \mid c_1,...c_n) = \prod_{i=1}^{n} p(t_i \mid h_i) \qquad \text{(iii)}$$

in which the conditional probability for each POC tag *t* given its history *h* is calculated as

$$p(t \mid h) = \frac{p(h,t)}{\sum_{t' \in T} P(h,t')} \qquad \text{(iv)}$$

2.3 The transformation-based tagger

The error-driven transformation-based tagger we used in this paper is Brill's POS tagger (1994) with minimal modification. The way this tagger is set up makes it easy for it to work in conjunction with other taggers. When it is used for its original task of POS tagging, the model is trained in two phases. In the first phase lexical information, such as the affixes of a word, is learned to predict POS tags. The rules learned in this phase are then applied to the training corpus. In the second phase, contextual information is learned to correct the wrong tags produced in the

first phase. In the segmentation task, since we are dealing with single characters, by definition there is no lexical information as such. Instead, the training data are first POC-tagged by the maximum entropy model and then used by the error-driven transformation-based model to learn the contextual rules. The error-driven transformation-based model learns a ranked set of rules by comparing the perfectly POC-tagged corpus (the reference corpus) with the same corpus tagged by the maximum entropy model (the maxent-tagged corpus). At each iteration, this model tries to find the rule that achieves the maximum gain if it is applied. The rule with the maximum gain is the one that makes the maxent-tagged corpus most like the reference corpus. The maximum gain is calculated with an evaluation function which quantifies the gain and takes the largest value. The rules are instantiations of a set of pre-defined rule templates. After the rule with the maximum gain is found, it is applied to the maxent-tagged corpus, which will better resemble the reference corpus as a result. This process is repeated until the maximum gain drops below a pre-defined threshold, which indicates improvement achieved through further training will no longer be significant. The training will then be terminated. The rule templates are the same as those used in Brill (1994), except that these rule templates are now defined over characters rather than words.

(4) Rule templates used to learn contextual information:

Change tag *a* to tag *b* when:

a. The preceding (following) character is tagged *z*.

b. The character two before (after) is tagged *z*.

c. One of the two preceding (following) characters is tagged *z*.

d. One of the three preceding (following) characters is tagged *z*.

e. The preceding character is tagged *z* and the following character is tagged *w*.

f. The preceding (following) character is tagged *z* and the character two before (after) was tagged *w*.

g. The preceding (following) character is *c*.

h. The character two before (after) is *c*.

i. One of the two preceding (following) characters is *c*.

j. The current character is *c* and the preceding (following) character is x.

k. The current character is *c* and the preceding (following) character is tagged *z*.

where *a*, *b*, *z* and *w* are variables over the set of four tags (LL, RR, LR, MM)

The ranked set of rules learned in this training process will be applied to the output of the maximum entropy tagger.

3 Experimental results

We conducted three experiments. In the first experiment, we used the maximum matching algorithm to establish a baseline, as comparing results across different data sources can be difficult. This experiment is also designed to demonstrate that even with a relatively small number of new words in the testing data, the segmentation accuracy drops sharply. In the second experiment, we applied the maximum entropy model to the problem of Chinese word segmentation. The results will show that this approach alone outperforms the state-of-the-art results reported in previous work in supervised machine-learning approaches. In the third experiment we combined the maximum entropy model with the error-driven transformation-based model. We used the error-driven transformation-based model to learn a set of rules to correct the errors produced by the maximum entropy model. The data we used are from the Penn Chinese Treebank (Xia *et al.* 2000, Xue *et al*. 2002) and they consist of Xinhua newswire articles. We took 250,389 words (426,292 characters or *hanzi*) worth of manually segmented data and divided them into two chunks. The first chunk has 237,791 words (404,680 Chinese characters) and is used as training data. The second chunk has 12,598 words (21,612 characters) and is used as testing data. These data are used in all three of our experiments.

3.1 Experiment One

In this experiment, we conducted two sub-experiments. In the first sub-experiment, we used a forward maximum matching algorithm to segment the testing data with a dictionary compiled from the training data. There are 497 (or 3.95%) new words (words that are not found in the training data) in the testing data. In the second sub-experiment, we used the same algorithm to segment the same testing data with a dictionary that was compiled from BOTH the training data and the testing data, so that there are no "new" words in the testing data.

3.2 Experiment Two

In this experiment, a maximum entropy model was trained on a POC-tagged corpus derived from the training data described above. In the testing phase, the sentences in the testing data were first split into sequences of characters and then tagged this maximum entropy tagger. The tagged testing data are then converted back into word segments for evaluation. Note that converting a POC-tagged corpus into a segmented corpus is not entirely straightforward when inconsistent tagging occurs. For example it is possible that the tagger assigns a LL-LR sequence to two adjacent characters. We made no effort to ensure the best possible conversion. The character that is POC-tagged LL is invariably combined with the following character, no matter how it is tagged.

3.3 Experiment Three

In this experiment, we used the maximum entropy model trained in experiment two to automatically tag the training data. The training accuracy of the maximum entropy model is 97.54% in terms of the number of characters tagged correctly and there are 9940 incorrectly tagged characters, out of 404,680 characters in total. We then used this output and the correctly tagged data derived from the manually segmented training data (as the reference corpus) to learn a set of transformation rules. 214 rules were learned in this phase. These 214 rules were then used to correct the errors of the testing data that was first tagged by maximum entropy model in experiment two. As a final step, the tagged and corrected testing data were converted into word segments. Again, no effort was made to optimize the segmentation accuracy during the conversion.

3.4 Evaluation

In evaluating our model, we calculated both the tagging accuracy and segmentation accuracy. The calculation of the tagging accuracy is straightforward. It is simply the total number of correctly POC-tagged characters divided by the total number of characters. In evaluating segmentation accuracy, we used three measures: precision, recall and balanced F-score. Precision (p) is defined as the number of correctly segmented words divided by the total number of words in the automatically segmented corpus. Recall (r) is defined as the number of correctly segmented words divided by the total number of words in the gold standard, which is the manually annotated corpus. F-score (f) is defined as follows:

$$f = \frac{p \times r \times 2}{p + r} \qquad \text{(v)}$$

The results of the three experiments are tabulated as follows:

tagger	tagging accuracy		segmentation accuracy		
	training	testing	testing		
			p(%)	r(%)	f(%)
1	n/a	n/a	87.34	92.34	89.77
2	n/a	n/a	94.51	95.80	95.15
3	97.55	95.95	94.90	94.88	94.89
4	97.81	96.07	95.21	95.13	95.17

Table 1

1 = maximum matching algorithm applied to testing data with new words
2 = maximum matching algorithm applied to testing data without new words
3 = maximum entropy tagger

4 = maximum entropy tagger combined with the transformation-based tagger

4 Discussion

The results from Experiment one show that the accuracy of the maximum matching algorithm degrades sharply when there are new words in the testing data, even when there is only a small proportion of them. Assuming an ideal scenario where there are no new words in the testing data, the maximum matching algorithm achieves an F-score of 95.15%. However, when there are new words (words not found in the training data), the accuracy drops to only 89.77% in F-score. In contrast, the maximum entropy tagger achieves an accuracy of 94.89% measured by the balanced F-score even when there are new words in the testing data. This result is only slightly lower than the 95.15% that the maximum matching algorithm achieved when there are no new words. The transformation-based tagger improves the tagging accuracy by 0.12% from 95.95% to 96.07%. The segmentation accuracy jumps to 95.17% (F-score) from 94.89%, an increase of 0.28%. That fact that the improvement in segmentation accuracy is higher than the improvement in tagging accuracy shows that the transformation-based tagger is able to correct some of the inconsistent tagging errors produced by the maximum entropy tagger. This is clearly demonstrated in the five highest-ranked transformation rules learned by this model:

(5) Top five transformation rules

RR MM NEXTTAG RR
LL LR NEXTTAG LL
LL LR NEXTTAG LR
MM RR NEXTBIGRAM LR LR
RR LR PREVBIGRAM RR LR

For example, the first rule says that if the next character is tagged RR, then change the current tag to MM from RR, since an RR RR sequence is inconsistent.

Incidentally, the combined segmentation accuracy is almost the same as that of the maximum matching method when there are no new words.

Evaluating this approach against previous results can be a tricky matter. There are several reasons for this. One is that the source of data can affect the segmentation accuracy. Since the results of machine-learning approaches are heavily dependent on the type of training data, comparison of segmenters trained on different data is not exactly valid. The second reason is that the amount of training data also affects the accuracy of segmenters. Still some preliminary observations can be made in this regard. Our accuracy is much higher that those reported in Hockenmaier and Brew (1998) and Xue (2001), who used error-driven transformation-based learning to learn a set of n-gram rules to do a series of merge and split operations on data from Xinhua news, the same data source as ours. The results they reported are 87.9% (trained on 100,000 words) and 90.2% (trained on 80,000 words) respectively, measured by the balanced F-score.

Using a statistical model called prediction by partial matching (PPM), Teahan *et al* (2000) reported a significantly better result. The model was trained on a million words from Guo Jin's Mandarin Chinese PH corpus and tested on five 500-segment files. The reported F-scores are in a range between 89.4% and 98.6%, averaging 94.4%. Since the data are also from Xinhua newswire, some comparison can be made between our results and this model. With less training data, our results are slightly higher (by 0.48%) when using just the maximum entropy model. When this model is combined with the error-driven transformation-based learning model, our accuracy is higher by 0.77%. Still, this comparison is just preliminary since different segmentation standards can also affect segmentation accuracy.

5 Conclusion

The preliminary results show that our approach is more robust than the dictionary-based approaches. They also show that the present

approach outperforms other state-of-the-art machine-learning models. We can also conclude that the maximum entropy model is a promising supervised machine learning alternative that can be effectively applied to Chinese word segmentation.

6 Acknowledgement

This research was funded by DARPA N66001-00-1-8915. We gratefully acknowledge comments from two anonymous reviewers.

7 References

Eric Brill. 1995. Some Advances In Rule-Based Part of Speech Tagging, *AAAI* 1994

Julia Hockenmaier and Chris Brew. 1998. Error-driven segmentation of Chinese. *Communications of COLIPS*, 1:1:69-84.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Part-of-Speech Tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, May 17-18, 1996. University of Pennsylvania.

W. J. Teahan, Rodger McNab, Yingying Wen and Ian H. Witten. 2000. A Compression-based Algorithm for Chinese Word Segmentation. *Computational Linguistics*, 26:3:375-393

Andi Wu and Zixin Jiang. 1998. Word Segmentation in Sentence Analysis. In *Proceedings of the 1998 International Conference on Chinese Information Processing*. Nov. 1998, Beijing, pp. 167-180.

Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Shizhe Huang, Tony Kroch, Mitch Marcus. 2000. Developing Guidelines and Ensuring Consistency for Chinese Text Annotation. In *Proc. of the 2nd International Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

Nianwen Xue. 2001. *Defining and Automatically Identifying Words in Chinese*. PhD Dissertation, University of Delaware.

Nianwen Xue, Fu-dong Chiou, Martha Palmer. 2002. Building a Large Annotated Chinese Corpus. To appear in *Proceedings of the 19th International Conference on Computational Linguistics*. August 14 - September 1, 2002. Taipei, Taiwan.