Maximum Entropy Word Segmentation of Chinese Text

Aaron J. Jacobs
Department of Linguistics
University of Texas at Austin
1 University Station B5100
Austin, TX 78712-0198 USA
aaronjacobs@mail.utexas.edu

Yuk Wah Wong
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233 USA
ywwong@cs.utexas.edu

Abstract

We extended the work of Low, Ng, and Guo (2005) to create a Chinese word segmentation system based upon a maximum entropy statistical model. This system was entered into the Third International Chinese Language Processing Bakeoff and evaluated on all four corpora in their respective open tracks. Our system achieved the highest F-score for the UPUC corpus, and the second, third, and seventh highest for CKIP, CITYU, and MSRA respectively. Later testing with the gold-standard data revealed that while the additions we made to Low et al.'s system helped our results for the 2005 data with which we experimented during development, a number of them actually hurt our scores for this year's corpora.

1 Segmenter

Our Chinese word segmenter is a modification of the system described by Low et al. (2005), which they entered in the 2005 Second International Chinese Word Segmentation Bakeoff. It uses a maximum entropy (Ratnaparkhi, 1998) model which is trained on the training corpora provided for this year's bakeoff. The maximum entropy framework used is the Python interface of Zhang Le's maximum entropy modeling toolkit (Zhang, 2004).

1.1 Properties in common with Low et al.

As with the system of Low et al., our system treats the word segmentation problem as a tagging problem. When segmenting a string of Chinese text, each character can be assigned one of four boundary tags: *S* for a character that stands alone as a word, *B* for a character that begins a multi-character word, *M* for a character in a multi-character word which neither starts nor ends the word, and *E* for a character that ends a multi-character word. The optimal tag for a given character is chosen based on features derived from the character's surrounding context in accordance with the decoding algorithm (see Section 1.2).

All of the feature templates of Low et al.'s system are utilized in our own (with a few slight modifications):

1. C_n $(n = -2, -1, 0, 1, 2)$
2. C_nC_{n+1} $(n = -2, -1, 0, 1)$
3. $C_{-1}C_1$
4. $Pu(C_0)$
5. $T(C_{-2})T(C_{-1})T(C_0)T(C_1)T(C_2)$
6. Lt_0
7. C_nt_0 $(n = -1, 0, 1)$

In the above feature templates, C_i refers to the character i positions away from the character under consideration, where negative values indicate characters to the left of the present position. The punctuation feature $Pu(C_0)$ is added only if the current character is a punctuation mark, and the function T maps characters to various numbers representing classes of characters. In addition to the numeral, date word, and English letter classes of Low et al.'s system, we added classes for punctuation and likely decimal points (which are defined by a period or the character 点 occurring between two numerals). L is defined to be the length of the longest word W in the dictionary that matches some sequence of characters around C_0

in the current context and t_0 is the boundary tag of C_0 in W. The dictionary features are derived from the use of the same online dictionary from Peking University that was used by Low et al.

In order to improve out-of-vocabulary (OOV) recall rates, we followed the same procedure as Low et al.'s system in using the other three training corpora as additional training material when training a model for a particular corpus:

1. Train a model normally using the given corpus.
2. Use the resulting model to segment the other training corpora from this year's bakeoff, ignoring the pre-existing segmentation.
3. Let C be a character in one of the other corpora D. If C is assigned a tag t by the model with probability p, t is equivalent to the tag assigned by the actual training corpus D, and p is less than 0.8, then add C (along with its associated features) as additional training material.
4. Train a new model using all of the original training data along with the new data derived from the other corpora as described in the previous step.

This procedure was carried out when training models for all of the corpora except CKIP. The model for that corpus was trained solely with its own training data due to time and memory concerns as well as the fact that our scores during development for the corresponding corpus (AS) in 2005 did not seem to benefit from the addition of data from the other corpora.

We adopt the same post-processing step as Low et al.'s system: after segmenting a body of text, any sequence of 2 to 6 words whose total length is at least 3 characters and whose concatenation is found as a single word elsewhere in the segmenter's output is joined together into that single word. Empirical testing showed that this process was actually detrimental to results in the 2005 CITYU data, so it was performed only for the UPUC, MSRA, and CKIP corpora.

1.2 Decoding algorithm

When segmenting text, to efficiently compute the most likely tag sequence our system uses the Viterbi algorithm (Viterbi, 1967). Only legal tag sequences are considered. This is accomplished by ignoring illegal state transitions (e.g. from a *B* tag to an *S* tag) during decoding. At each stage the likelihood of the current path is estimated by multiplying the likelihood of the path which it extends with the probability given by the model of the assumed tag occurring given the surrounding context and the current path. To keep the problem tractable, only the 30 most likely paths are kept at each stage.

The advantage of such an algorithm comes in its ability to 'look ahead' compared to a simpler algorithm which just chooses the most likely tag at each step and goes on. Such an algorithm is likely to run into situations where choosing the most likely tag for one character forces the choice of a very sub-optimal tag for a later character by making impossible the choice of the best tag (e.g. if *S* is the best choice but the tag assigned for the previous character was *B*). In contrast, the Viterbi algorithm entertains multiple possibilities for the tagging of each character, allowing it to choose a less likely tag now as a trade-off for a much more likely tag later.

1.3 Other outcome-independent features

To the feature templates of Low et al.'s system described in Section 1.1, we added the following three features which do not depend on previous tagging decisions but only on the current character's context within the sentence:

1. The `surname` feature is set if the current character is in our list of common surname characters, as derived from the Peking University dictionary.
2. The `redup-next` feature is set if C_1 is equal to C_0. This is to handle reduplication within words, such as in the case of 清清楚楚 'particularly clear'.
3. The `redup-prev` feature is set if C_{-1} is equal to C_0.

These features were designed to give the system hints in cases where we saw it make frequent errors in the 2005 data.

1.4 Outcome-dependent features

In addition to the features previously discussed, we added a number of features to our system that are *outcome-dependent* in the sense that their realization for a given character depends upon how the previous characters were segmented. These work in conjunction with the Viterbi algorithm discussed in Section 1.2 to make it so that a given character in a sentence can be assigned a different set of features each time it is considered, depending on the path currently being extended.

1. If the current character is one of the place characters such as 村 or 镇 which commonly occur at the end of a three-character word and the length of the current word (as determined by previous tagging decisions on the current path) including the current character is equal to three, then the feature `place-char-and-len-3` is set.

2. If the situation is as described above except the *next* character in the current context is the place character, then the feature `next-place-char-and-len-2` is set.

3. If the current character is 等 and the word before the previous word is an enumerating comma (、), then the feature `deng-list` is set. This is intended to capture situations where a list of single-word items is presented, followed by 等 to mean 'and so on'.

4. If the current character is 等 and the third word back is an enumerating comma, then the feature `double-word-deng-list` is set.

5. If the length of the previous word is at least 2 and is equal to the length of the current word, then the feature `symmetry` is set.

6. If the length of the previous word is at least 2 and is one more than the length of the current word, then the feature `almost-symmetry` is set.

7. Similar features are added if the length of the current word is equal to (or one less than) the length of the word before the last and the last word is a comma.

These features were largely designed to help alleviate problems the model had with situations in which it would otherwise be difficult to discern the correct segmentation. For example, in one development data set the model incorrectly grouped 等 at the end of a list (which should be a word on its own) with the following character to form 等同, a word found in the dictionary.

1.5 Simplified normalization

To derive the most benefit from the additional training data obtained as described in Section 1.1, before generating any sort of features from characters in training and test data, all characters are normalized by the system to their simplified variants (if any) using data from version 4.1.0 of the Unicode Standard. This is intended to improve the utility of additional data from the traditional Chinese corpora when training models for the simplified corpora, and vice versa. Due to the results of some empirical testing, this normalization was only performed when training models for the UPUC and MSRA corpora; in our testing it did not actually help with the scores for the traditional Chinese corpora.

2 Results

Table 1 lists our official results for the bakeoff. The columns show F scores, recall rates, precision rates, and recall rates on out-of-vocabulary and in-vocabulary words. Out of the participants in the bakeoff whose scores were reported, our system achieved the highest F score for UPUC, the second-highest for CKIP, the seventh-highest for MSRA, and the third-highest for CITYU.

Corpus	F	R	P	R_{OOV}	R_{IV}
UPUC	0.944	0.949	0.939	0.768	0.966
CKIP	0.954	0.959	0.949	0.672	0.972
MSRA	0.960	0.959	0.961	0.711	0.968
CITYU	0.969	0.971	0.967	0.795	0.978

Table 1: Our 2006 SIGHAN bakeoff results.

The system's F score for MSRA was higher than for UPUC or CKIP, but it did particularly poorly compared to the rest of the contestants when one considers how well it performed for the other corpora. An analysis of the gold-standard files for the MSRA test data show that out of all of the corpora, MSRA had the highest percentage of single-character words and the smallest percentage of two-character and three-character words. Moreover, its proportion of words over 5 characters in length was five times that of the other corpora. Most of the errors our system made on the MSRA test set involved incorrect groupings of true single-character words. Another comparatively high proportion involved very long words, especially names with internal syntactic structure (e.g. 中国国民党革命委员会第九次全国代表大会).

Our out of vocabulary scores were fairly high for all of the corpora, coming in first, fourth, fifth, and third places in UPUC, CKIP, MSRA, and CITYU respectively. Much of this can be attributed to the value of using an external dictionary and additional training data, as illustrated by the experiments run by Low et al. (2005) with their model.

3 Further testing

In order to get some idea of how each of our additions to Low et al.'s system contributed to our results, we ran a number of experiments with the gold-standard segmentations distributed after the completion of the bakeoff. We stripped out all of the additions and then added them back in one by one, segmenting and scoring the test data each time. What we found is that our system actually performed best with the implementation of the Viterbi algorithm (which raised F scores by an average of about 0.09 compared to simply choosing the most likely tag at each stage) but without any of the extra outcome-dependent or independent features. There were only two exceptions to this:

- The system achieved slightly higher OOV recall rates for the MSRA and CITYU corpora with the `place-char` and `deng-list` features than without.
- The system achieved a very small increase in F score for the UPUC corpus with the `place-char` feature than without.

Besides these small differences, the model was best off without any of the features enumerated in Sections 1.3 and 1.4, obtaining the scores listed in Table 2. This is a surprising result, as in our testing the added features helped to improve the F scores and OOV recall rates of the system when dealing with the 2005 bakeoff data, even if only by a small amount in some cases.

It should be noted that in our testing during development, even when we strove to create a system which matched as closely as possible the one described by Low et al. (2005), we were unable to achieve scores for the 2005 bakeoff data as high as their system did. Why this was the case remains a mystery to us. It is possible that at least some of the gap is due to implementation differences. In particular, the maximum entropy toolkit utilized along with the training algorithms chosen seem likely candidates for sources of the disparity.

Corpus	F	R	P	R_{OOV}	R_{IV}
UPUC	0.948	0.954	0.943	0.781	0.970
CKIP	0.957	0.962	0.952	0.698	0.973
MSRA	0.964	0.963	0.964	0.731	0.971
CITYU	0.974	0.976	0.972	0.816	0.983

Table 2: Our results without the extra features.

4 Conclusions

Using a maximum entropy approach based on a modification of the system described by Low, Ng, and Guo (2005), our system was able to achieve a respectable level of accuracy when evaluated on the corpora of the word segmentation task of the Third International Chinese Language Processing Bakeoff. Implementing the Viterbi decoding algorithm was very beneficial for F scores and OOV recall rates. However, it should be investigated whether the rest of the added features, especially the outcome-dependent ones, are useful in general or if they were only beneficial for the 2005 test data due to some pattern in that data, after which they were modeled.

References

Jin Kiat Low, Hwee Tou Ng, and Wenyuan Guo. 2005. A maximum entropy approach to Chinese word segmentation. In *Fourth SIGHAN Workshop on Chinese Language Processing*, pages 161–164. URL `http://www.aclweb.org/anthology/I05-3025`.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Le Zhang, 2004. *Maximum Entropy Modeling Toolkit for Python and C++*. URL `http://homepages.inf.ed.ac.uk/s0450736/`.