Combining Language Modeling and Discriminative Classification for Word Segmentation

Dekang Lin

Google, Inc.
1600 Amphitheater Parkway, Mountain View, CA, USA, 94043
lindek@google.com

Abstract. Generative language modeling and discriminative classification are two main techniques for Chinese word segmentation. Most previous methods have adopted one of the techniques. We present a hybrid model that combines the disambiguation power of language modeling and the ability of discriminative classifiers to deal with out-of-vocabulary words. We show that the combined model achieves 9% error reduction over the discriminative classifier alone.

Keywords: Segmentation, Maximum Entropy, Language Model.

1 Introduction

The problem of word segmentation is to identify word boundaries in languages, such as Chinese, Japanese and Thai, where such boundaries are not explicitly marked with white spaces. Word segmentation is the first step in processing the text in these languages and its quality often affects all downstream components.

The main two challenges for word segmentation are the resolution of ambiguities and the handling of out-of-vocabulary (OOV) words.

Ambiguity is pervasive in word segmentation. Consider the following fragment of a Chinese sentence:

…中国家鼓励… (in … the country encourages …) (1)

The correct segmentation is:

中	国家	鼓励
in	country	encourages

However, another (incorrect) candidate is:

中国	家	鼓励
China	home	encourages

An obvious way to resolve the ambiguities is through language modeling. The best segmentation of an input sentence $C_1^n = C_1C_2...C_n$ is the one where the resulting sequence of words has the highest probability among all word sequences that constitute the character sequence. In other words, the best segmentation is

A. Gelbukh (Ed.): CICLing 2009, LNCS 5449, pp. 170–182, 2009.

$$\arg\max_{yield(W_1^m)=C_1^n} P(W_1^m) \tag{2}$$

where $yield(W_1^m)$ is the concatenation of the characters in the word sequence.

Although a language model helps resolve segmentation ambiguities, the search for the best segmentation is limited to sequences of known words. The out-of-vocabulary (OOV) words are segmented into smaller units or missegmented with adjacent characters. To deal with the OOV problem, language model based segmenters typically rely on manually created pattern matching rules to recognize names, or heuristic post-processing rules to combine sequences of single characters.

Word segmentation can also be treated as a tagging problem (Xue and Shen, 2003; Peng *et. al.* 2004; Tseng *et al.* 2005; Low *et al.* 2005). Each character in the input sentence is assigned a tag. The best segmentation is determined by

$$\arg\max_{T_1^n} P(T_1^n | C_1^n) \tag{3}$$

where T_i is a tag that marks whether the character C_i is the beginning of a word. In the literature, two types of tags have been proposed. One is a 2-tag set: *b* for beginning of a word, and *c* for continuation of a word. The other is a 4-tag set:

s: the character is a word itself.
b: the character is the first character of a word.
e: the character is the last character of a word.
m: the character is in the middle of a word.

The probability in (3) is typically computed by a discriminative classifier trained with the maximum entropy and conditional random fields.

The word boundaries determined by the tags do not have to agree with any vocabulary, which means that the segmenter is able to produce words that it has never seen before. This turns out to be both a blessing and a curse. It is a blessing because such a segmenter combines new word recognition and segmentation in a single unified process. For example, the machine learned classifier may pick up the fact that the character 者 (suffix -er) is typically the last character in a word, and recognize购彩者 (lotto-ticket buyer) as a single word even if the word is never seen in the training corpus.

Not having to agree with a vocabulary may also be a curse because the segmenter may make a sequence of tagging decisions that are individually quite reasonable, but globally inconsistent. Consider the example in (1). Since 中国 (China) is a common word, the classifier is likely to say there is no boundary between中 and 国. Since国家 (country) is also very common, it decides that there is no boundary between 国 and 家 either. As a result, the segmenter incorrectly outputs 中国家as a single word. One might hope that models such as Maximum Entropy Markov Mode (MEMM) or Conditional Random Fields (CRF) (Lafferty *et al.* 2000) will be able to use the preceding tags as features to ensure the global consistency of a tag sequence. However, such hope gets quickly shattered when one realizes that the conditional distribution of tags given a tag n-gram is not nearly as sharp as the tag distribution given the observations. This phenomenon was first noted by Klein and Manning (2002) in part-of-speech tagging.

They call it the **observation bias**. In word segmentation, the bias is even stronger, because there are only 2-4 types of tags.

In this paper, we present a word segmenter that combines a discriminative classifier with a language model. It is constructed by first training the discriminative classifier with a manually segmented corpus and then building an n-gram language model smoothed with distributionally similar words. The word similarities are obtained from an unlabeled corpus segmented with the discriminative classifier.

The remainder of the paper is organized as follows. The next section gives an overview of our model. We then provide details of the discriminative classifier and the language model augmentation in Sections 3 and 4, respectively. After presenting the experimental results in Section 5, we discuss the related work in Section 6.

2 A Probabilistic Segmentation Model

Our segmentation model combines a discriminative classifier and a generative language model. Theoretically, (3) is equivalent to

$$\arg\max_{yield(W_1^m)=C_1^n} P(W_1^m | C_1^n) \tag{4}$$

The reason is that once the tags are known, the word sequence is determined. In practice, however, C_1^n is first converted into features in order for the probability computation to be feasible. The features are represented as a set of atomic identifiers. The probability models, such as Maximum Entropy or CRF classifier, do not really have access to the character sequence C_1^n. We propose a different formulation of the word segmentation problem. To segment a sentence is to find

$$\arg\max_{yield(W_1^m)=C_1^n} P(W_1^m | F_1^n) \tag{5}$$

where F_1^n is a sequence of feature vectors, each of which corresponds to a character in the input sentence. The probability $P(W_1^m | F_1^n)$ is computed as follows:

$$P(W_1^m | F_1^n) = P(W_1^m, T_1^n | F_1^n) \tag{6}$$

$$= P(W_1^m | T_1^n, F_1^n) \times P(T_1^n | F_1^n) \tag{7}$$

$$\approx P(W_1^m | T_1^n) \times P(T_1^n | F_1^n) \tag{8}$$

$$\approx \prod_{i=1}^{m} P(W_i | W_{i-k}^{i-1}) \times \prod_{i=1}^{n} P(T_i | F_i) \tag{9}$$

The equality in 0 holds because the tag sequence T_1^n is logically implied by the word sequence W_1^n. The approximation in 0 is based on the assumption that if tags are known, the feature vectors are conditionally independent of the word sequence. In step 0 we assume that the probabilities of tags are independent of each other (except that their corresponding feature vectors are correlated) and the probability of a word

given all previous words is equal to the probability of the word given its previous k–1 words.

Therefore, our segmentation model is a combination of a generative k-gram language model $P\left(W_i \middle| W_{i-k}^{i-1}\right)$ and a discriminative classifier $P\left(T_i \middle| F_i\right)$.

The discriminative classifier can be used as a segmenter by itself. We will refer to it as **M1**. We first train M1 with a maximum entropy learner using a segmented corpus. We then use M1 to segment an unlabeled corpus to acquire OOV words and build a language model. The combination of the language model and the maximum entropy classifier results in a better segmenter, which will be referred to as **M2**.

3 M1: A Discriminative Classifier

Our word-boundary classifier is similar to the maximum entropy segmenter in (Low *et al.*, 2005). For each character in the segmented training corpus, we extract a training example consisting of a label (one of *s*, *b*, *e*, or *m*) and a feature vector.

3.1 Features

The features are extracted using the following templates (the numbers –1, 0, 1 refer to the previous, the current and the next character position respectively):

C_i (i = –1, 0, 1): the character unigrams
C_iC_{i+1} (–1, 0): the character bigrams

W_{kw}: true when a known word w is found in the input sentence and the position of the current character is k relative to w. The value of k ranges from –1 to |w|, the number of characters in w. When k = –1, the word w is immediately after the current character. When k = |w|, w immediately precedes the current character. Otherwise, the current character is part of w.

For example, the following features are extracted for the character 家 in a sentence fragment 中国家鼓励.

$C_{0家}$, $C_{-1国}$, $C_{1鼓}$, $C_{-10国家}$, $C_{01家鼓}$, $W_{-1鼓励}$, $W_{1国家}$, $W_{2中国}$

Our feature set differs from previous approaches in several aspects.

The feature template W includes both overlapping and adjacent word candidates. The overlapping words were used in (Low *et al.*, 2005). None of the previous segmenters seems to have used the adjacent words as features.

We extracted character bigrams as features from the 2 character window [–1, 1]. Most other systems used the window [–2, 2]. The narrower window reduces the number of features. Our experiments show that smaller window size does not compromise the accuracy. Note that, with the overlapping and adjacent words, the segmenter does consider characters more than one character away, as long as there exists a known word connecting them to the current character.

The word features in our model are triggered by the presence of a sequence of characters that form a known word. This is different from (Andrew 2006), where the semi-CRF features are keyed on words that are guaranteed to be part of the final

segmentation. For example, in the above example, the presence of the features $W_{1国家}$ and $W_{2中国}$ does not guarantee the words 国家 (country) and中国 (China) to be in the output. In fact, the output may include neither of the words.

3.2 MaxEnt Training

After extracting the tags and feature vectors from a training corpus consisting of segmented sentences, we trained a maximum entropy classifier with the GIS algorithm using exponential prior for regularization (Goodman 2004).

Let (x_j, y_j) denote a training example with feature vector x_j and label y_j, f_i (i=1…F) denote a feature, and $\Lambda = (\lambda_1, \lambda_2, ..., \lambda_F)$ denote the weight vector corresponding to the features. The training algorithm maximizes

$$\arg\max_{\Lambda} \prod_{j=1}^{n} \frac{\exp \sum_{i=1}^{F} \lambda_i f_i(x_j, y_j)}{\sum_{y'} \exp \sum_{i=1}^{F} \lambda_i f_i(x_j, y')} \times \prod_{i=1}^{F} \alpha_i \exp(-\alpha_i \lambda_i)$$

subject to the constraint that

$\lambda_i = 0$ and expected[f_i] ≥ observed[f_i] – α_i, or

$\lambda_i > 0$ and expected[f_i] = observed[f_i] – α_i.

We found that the performance of the trained segmenter is not very sensitive to the value of α_i as long as it is within (0, 1). We therefore used the same value, 0.1, for all features in all of our experiments. The number of training iterations was fixed at 300 in all experiments as well.

Unlike (Low *et al.*, 2005), we did not set a minimum threshold for feature counts. All the features extracted from the training examples are used. A nice property of the exponential prior, as pointed out by (Goodman, 2004), is that the weights of many features are 0 after training and can be discarded. The MaxEnt training algorithm is therefore capable of doing feature selection by itself.

3.3 Consistent Tag Sequences

A problem with the *s-b-e-m* tag set is that not all tag sequences are consistent. For example, *s* cannot be followed by *e* or *m*. The issue was addressed in (Xue and Shen 2003) by applying transformation-based learning to the output tag sequences. Another solution in (Low *et al.*, 2005) is to restrict the word lengths to a predefined number, such as 20, and then use a dynamic programming algorithm to find most probably tag sequences with consistent tags.

We adopted a dynamic programming solution that is more principled (no predefined limit) than (Low *et al.*, 2005). We represent each possible tag for each character as a node in a Markov network. The emission probability of a node is the probability of the tag obtained from the MaxEnt classifier. The transition probabilities are all 1s.

However, transitions only exist if the tag combination is valid. For example, the *s* node of a character has only transitions to the *s* and *b* nodes of the next character. The most probable consistent tag sequence can then be obtained with the Viterbi algorithm.

The resulting MaxEnt classifier can be used as a segmenter itself. Even though it doesn't resort to special word lists for recognizing names or OOV words and doesn't have any post processing as many of the top systems in the Chinese Segmentation Bakeoff, its performance is surprisingly good, in fact, better than or equal to any system in the Second Chinese Segmentation Bakeoff on all 4 data sets tested there (see Section 5 for details).

4 M2 = M1 + Language Modeling

The segmenter M2 is built by adding a language model to M1. There are several options for building the language model component. One is to collect the n-gram statistics from the segmented training corpus. The problem is that the segmented corpus is small and the n-gram statistics is consequently very sparse. Another option is to run the segmenter on a large corpus and gather the statistics from the automatically segmented corpus. This type of self-training has been tried on many other NLP tasks, including parsing (Charniak 1997) and part of speech tagging (Clark *et al.*, 2003). The results were mostly disappointing, with the exception of (McClosky *et. al.* 2006). The self-trained system often perpetuates and sometimes amplifies the errors in the original system. We propose a third alternative, which is to collect n-gram counts from the manually segmented corpus and smoothing the counts with distributional word similarities obtained from an unsegmented corpus. The advantage of this is that we are leveraging on the second order regularities in the corpus. Even if there are systematic errors in the n-gram counts, there has to be systematic such errors to cause the distributional similarity to be erroneous.

4.1 Distributional Similarity

Distributional word similarities are computed under the assumption that words that occur in similar contexts tend to have similar meanings (Hindle 1990; Dagan *et al.*, 1997; Lin 98). We represent the contexts of words as feature vectors. We define a feature *f* of a word *w* to be a word that occurred next to *w* on its immediate left or right. The value of the feature *f* is the point-wise mutual information between *f* and the word *w*:

$$PMI(w,f) = \log \frac{P(w,f)}{P(w) \times P(f)} = \log \frac{(|w,f| - d) \times |*,*|}{|w,*| \times |*,f|}$$

where $|w, f|$ is the frequency count of the co-occurrence of *w* and *f*, and * is a wild card, and *d* is a discounting factor (we used $d = 0.9$ in out experiments). The similarity between two words can then be computed by the cosine of the angle between their respective feature vectors.

4.2 OOV Word Extraction

The language model needs to work with a closed vocabulary. The vocabulary of the segmented training corpus is small (the training corpora in the Chinese Segmentation Bakeoff contain 55k-140k unique words). Fortunately, by running the discriminative classifier M1 on a large corpus, we may obtain a large number of 'words' that are not part of the training vocabulary. However, many of the 'words' are segmentation errors. We assume that the vocabulary of the training corpus is sufficiently large so that, for any word, there exists a set of words in the vocabulary that are distributionally similar to it. Based on this assumption, we constructed feature vectors for all words that are sufficiently frequent in the automatically segmented corpus. For each OOV word, we compute its similarity with all of the in-vocabulary words. A word candidate is discarded if it is not distributionally similar (based on a threshold) to any known word.

4.3 Similarity-Based Smoothing

We use the word similarities to generalize the language model constructed with the segmented corpus. Similarity-based smoothing was first proposed in (Dagan *et. al.* 1997), where the bigram probability $P(w_2|w_1)$ is smoothed with:

$$P_{SIM}(w_2|w_1) = \sum_{w_1' \in S(w_1)} \frac{W(w_1, w_1')}{N(w_1)} P(w_2|w_1')$$

where $S(w_1)$ is the set of top similar words of w_1, $W(w_1, w_1')$ is the similarity between w_1 and w_1' and $N(w_1)$ is the normalization factor. The problem with this scheme is that since $P(\bullet|w_1')$ is likely to be a sparse distribution as well. Many of the probabilities in the sum are also 0s. We propose a different solution, by assuming that similar word pairs have similar values of PMI.

$$P(w_2|w_1) = \frac{P(w_1, w_2)}{P(w_1)} = \frac{P(w_1, w_2)}{P(w_1)P(w_2)} P(w_2) = \exp(PMI(w_1, w_2))P(w_2)$$

$$\approx \exp\left(\sum_{w_1' \in S(w_1), w_2' \in S(w_2)} \frac{W(w_1, w_2, w_1', w_2')}{N(w_1, w_2)} PMI(w_1', w_2') \right) P(w_2)$$

where $W(w_1, w_2, w_1', w_2')$ is the similarity between two pairs of words (w_1, w_2) and (w_1', w_2'), which is compute as the geometric average of the similarities between corresponding words. The advantage of this smoothing scheme is that either or both of w_1 and w_2 may be substituted with their similar words.

4.4 Combining LM with MaxEnt

The segmenter M2 employs a dynamic programming algorithm to find the best sequence of words that maximizes the combined probabilities of the language model and the tag sequence probability. The algorithm constructs a Markov network. Each

node in the network is a pair (w, l), where w is a word known to the language model and l is a label which is either B or C. When the label l is B (begin), w starts a new segment. When the label is C (continuation), w is appended to the end of a previous segment. The best segment sequence is obtained by finding a path with highest probability from the beginning to the end of the sentence. Each B-labeled node and zero or more C-labeled nodes that follow it form a segment. A segment with a single B node corresponds to a know word in the language model. Other segments are OOV words.

The emission probability of a node (w, l) is the product of the labels of the characters in w. If the label is B, the characters in w are labeled as s or $bm…e$, depending on the number of characters in w. If the label is C, the characters in w are labeled e or $m…e$.

There is a transition from (w_1, l_1) to (w_2, l_2) if w_1 is immediately followed by w_2 in the input. The probability of the transition is computed as follows:

- If $l_1 = B$ and $l_2 = B$, the transition probability is bigram probability $P(w_2 \mid w_1)$ computed by the language model.
- If $l_1 = C$ and $l_2 = B$, the transition probability is unigram probability $P(w_2)$.
- If $l_1 = B$ and $l_2 = C$, w_2 is appended to w_1 to form a longer segment. The emission probability of (w_1, l_1) was computed under the assumption that the last character of w_1 is tagged as e or s. When w_2 is appended to w_1, the tag of the last character of w_1 needs to be changed to m or b. We therefore compute the transition probability by multiplying $P(w_2)$ with the ratio between the probabilities of last character of w being m and being e (or being b and being s).
- If $l_1 = C$ and $l_2 = C$, the tag of the last character of w_1 needs to be changed from e to m. The transition probability is the unigram probability $P(w_2)$ multiplied with the ratio between the probabilities of the last character of w_1 being m and being e.

5 Experimental Results

5.1 Data Sets

There is no universally accepted standard for Chinese segmentation. The Second Chinese Segmenter Bakeoff (Emerson 2005) tested the systems with 4 sets of data, each of which was created according to its own guidelines. The systems are evaluated by the F-measure of how well they retrieve the gold standard segments in the test set.

Table 1 summarizes the general characteristics of the data sets. Each data set consists of a training set and a test set. The baseline is obtained by maximal matching using only the words from the training data. It can be seen that the OOV rate directly correlates with baseline F-measure.

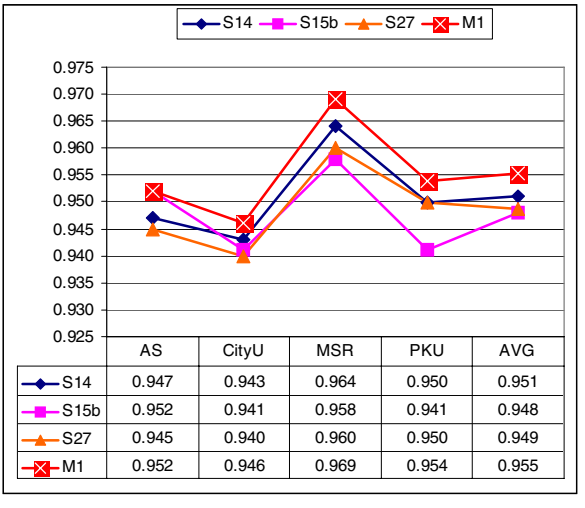
Table 1. Data Sets in the Second Chinese Segmentation Bakeoff

Data Set	Source	Chinese	Training (words)	Test (words)	OOV rate	Baseline
AS	Academia Sinica	trad.	5.45M	122K	4.3%	0.882
CityU	City University	trad.	1.46M	41K	7.4%	0.833
MSR	Microsoft	simp.	2.37M	107K	2.6%	0.933
PKU	Peking University	simp.	1.10M	104K	5.8%	0.869

Table 2. Comparison with alternative designs

	AS	CityU	MSR	PKU	AVG
M1	0.952	0.946	0.969	0.954	0.955
5-char	0.952	0.947	0.968	0.952	0.955
2-tag	0.945	0.936	0.962	0.946	0.947

There were two tracks in the Chinese Segmentation Bakeoff: closed and open. The closed track restricts the systems to use only the training data set provided to all the participants. In the open track, systems are allowed to include their own resources. Our segmenter M1 qualifies as a closed track system. We compare it with closed track systems.



	AS	CityU	MSR	PKU	AVG
S14	0.947	0.943	0.964	0.950	0.951
S15b	0.952	0.941	0.958	0.941	0.948
S27	0.945	0.940	0.960	0.950	0.949
M1	0.952	0.946	0.969	0.954	0.955

Fig. 1. Comparison with closed track winners

5.2 Evaluation of M1

Figure 1 compares M1 against the three systems with a highest score in at least one of the data sets. S14, S15b and S27 are identifiers assigned to the systems (Emerson 2005). M1 performed significantly better than the best systems in the closed track on three out of the four data sets and tied for the forth one. On average, M1 obtained an error reduction[1] of 8% compared to the S14, which had the highest average F-score among the closed track participants. This is a little surprising because:

- M1 extracts features from a narrower window than most other systems;
- M1 does not involve any language-specific features. In contrast, the top systems all included rules or features that are keyed on special character sets extracted from

[1] We use the term loosely by treating 1-F as the error rate.

the training data, e.g., the characters that tend to be suffixes or prefixes, or the characters that tend to be surname or given names.

- M1 does not include any post processing, as many top systems.
- The machine learning algorithm in M1 is simpler (MaxEnt vs. CRF) and is faster to train.

Since many other systems used a 5-character window for extracting features and 2-tag set (*b* and *c*), instead of the 4-tag set (*s-b-e-m*), Table 2 shows the F-scores of these alternatives. Expanding the feature window from 3 characters to 5 characters essentially has no impact on the F-score. The downside is that the learned models become twice as big (see Table 3) and are computationally more costly to run. The 2-tag models are smaller. The smaller size, unfortunately, comes with a big drop in F-score. Table 3 also included the number parameters in trained CRF models in (Tseng *et al.,* 2005). They are generally more than twice as big as M1 models.

Table 3. Number of parameters (in millions)

	AS	CityU	MSR	PKU
M1	7.1	3.2	4.2	2.5
5-char	14.2	6.4	8.4	4.9
2-tag	3.0	1.3	1.7	1.0
CRF	8.1	7.1	12.5	5.4

Table 4. OOV word extraction

Words	AS	CityU	MSR	PKU
candidates	497745	547143	2593349	2115040
results	170434	175855	661576	549242

5.3 Evaluation of M2

The language models in M2 are constructed by segmenting a 5GB traditional Chinese news corpus (for AS and CityU) and a 50GB simplified Chinese news corpus (for MSR and PKU) with a discriminatively trained M1. The 'candidates' row in Table 4 contains the numbers of words in the M1-segmented corpus with a minimum frequency count of 10. The numbers of words that passed the distributional similarity filter are listed in the 'results' row. The M1-segmenters trained with AS and MSR produced more words because these two corpora treat the concatenation of a family and a given name as a single segment.

Table 5 compares M2 against M1. $\mathrm{ER}_{\mathrm{M1}}^{\mathrm{M2}}$ is the relative error reduction of M2 over M1. The introduction of a language model gets a relative error reduction of 9% on average. The F-score increases on all 4 data sets, although the increase is not statistically significant with the AS data. The no-sim row contains the results obtained by using all new word candidates obtained by segmenting the unlabeled data. The average F-score remain unchanged compared to M1. The tinyLM row shows the F-scores obtained if we use the segmented training set to build an n-gram language model

without similarity based smoothing. The average F-score decreases significantly. The drop is rather dramatic for the AS data. This, we believe, is because the language model information in the training data may have already been captured by the MaxEnt model. Another way to view this is that having a separate language model component is a good thing. Even if the discriminative classifier can capture some of the language model information, it can only do so for word sequences that are present in the manually segmented corpus. Having a language model component allows us to obtain this information from much larger (practically unlimited) amount of data.

Table 5. Results with language modeling

	AS	CityU	MSR	PKU	AVG
M2	0.953	0.954	0.972	0.958	0.959
M1	0.952	0.946	0.969	0.954	0.955
ER_{M1}^{M2}	2%	15%	10%	9%	9%
no-sim	0.952	0.951	0.970	0.956	0.957
tinyLM	0.897	0.946	0.965	0.949	0.939

The rankings of M2 would have been 2nd (AS), 2nd (CityU), 1st (MSR) and 10th (PKU) in the open track of the Second Chinese Segmentation Bakeoff. A notable difference between M2 and the open track systems is that the only additional resource in M2 is an unsegmented corpus, whereas others added manually segmented corpora, larger dictionaries, specialized word lists (e.g., punctuation marks, function words of various kinds, last names, country names, title prefixes, units, dates, *etc.*). For example, many systems used a lexicon from PKU[2]. If we include that lexicon in M2 as well, the F-score for PKU data set increases to 0.965 and ranks the 4th.

6 Related Work and Discussion

Many early statistical segmenters are based on language modeling, e.g., (Sproat *et al.*, 1996) and (Luo and Roukos, 1996). More recently, discriminative classification has proven to be a very effective method for this problem. Most top systems in the Chinese Segmenter Bakeoff are based on discriminative classifiers (Low *et al.* 2005; Tseng *et al., 2005*), although the unigram language model system by (Chen *et al.*, 2005) is also one of the strongest contenders. Our experiments show that the *s-b-e-m* tag set works better than the *b-c* tag set. The main difference between them is that the former is able to distinguish single-character words, which include many frequent function words, from others. Our method for ensuring the consistency of the 4-tag-set sequences is more principled than previous proposals.

Generative language models and discriminative classifiers complement each other in segmentation. One is helpful in resolving ambiguities while the other is good at dealing with OOV words. (Gao *et al.* 2005) combines the scores from many

[2] http://ccl.pku.edu.cn/doubtfire/Course/Chinese%20Information%20Processing/Source_Code/Chapter_8/Lexicon_full_2000.zip

components, including a language model, as features in a linear classifier. Their language model consists of a sequence of *word classes*, such as person names, locations, *etc.*, whereas we model sequences of words directly. The word-class sequence model is also used in (Asahari *et al.,* 2003). They then employ a SVM classifier to make corrections to the LM segmenter.

Another hybrid model is the semi-Markov CRF (Andrew 2006) where features may be defined in terms bigrams of words, instead of character sequences. Such features, however, are restricted to the labeled training set, whereas our language model is built from a much larger unlabeled data. The semi-Markov CRF segmenter in (Andrew 2006) was evaluated with the MSR data set. Its F-score is 0.968, which is slightly below our M1's 0.969 and significantly lower than M2's 0.972.

7 Conclusion

Previously word segmenters mostly rely on one of the language modeling and discriminative classification. We presented a segmenter that combines both. Our experimental results showed that the combined model achieves 9% error reduction over the discriminative classifier alone.

References

1. Andrew, G.: A hybrid Markov/Semi-Markov conditional random field for sequence segmentation. In: Proc. of EMNLP 2006 (2006)
2. Asahara, M., Goh, C., Wang, X., Matsumoto, Y.: Combining segmenter and chunker for Chinese word segmentation. In: Proc. of the Second SIGHAN Workshop on Chinese Language Processing, pp. 144–147 (2003)
3. Charniak, E.: Statistical parsing with a context-free grammar and word statistics. In: Proc. of AAAI 1997 (1997)
4. Clark, S., Curran, J., Osborne, M.: Bootstraping POS-taggers using unlabelled data. In: Proc. of CoNLL 2003 (2003)
5. Chen, Y., Zhou, A., Zhang, G.: Unigram Language Model for Chinese Word Segmentation. In: Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing (2005)
6. Dagan, Lee, L., Pereira, F.C.N.: Similarity based methods for word sense disambiguation. In: Proc. of ACL 1997 (1997)
7. Emerson, T.: The Second International Chinese Word Segmentation Bakeoff. In: Proc. SIGHAN Workshop on Chinese Language Processing (2005)
8. Gao, J., Li, M., Wu, A., Huang, C.N.: Chinese Word Segmentation and Named Entity Recognition: A Pragmatic Approach. Computational Linguistics 31(4), 531–574 (2005)
9. Goodman, J.: Exponential Priors for Maximum Entropy Models. In: Proceedings of HLT/NAACL (2004)
10. Hindle, D.: Noun classification from predicate-argument structures. In: Proc. of ACL 1990 (1990)
11. Klein, D., Manning, C.: A Generative constituent-context model for improved grammar induction. In: Proceedings of the 40th Annual Meeting of the ACL (2002)

12. Low, J.K., Ng, H.T., Guo, W.: A maximum entropy approach to Chinese word segmentation. In: Proc. SIGHAN Workshop on Chinese Language Processing (2005)
13. Lin, D.: Automatic retrieval and clustering of similar words. In: Proc. of COLING/ACL 1998, pp. 768–774 (1998)
14. Luo, X., Roukos, S.: An Iterative Algorithm to Build Chinese Language Models. In: Proc. of ACL 1996, pp. 139–145 (1996)
15. Luo, X.: A maximum entropy Chinese character-based parser. In: Proc. of EMNLP (2003)
16. McClosky, D., Charniak, E., Johnson, M.: Effective self-training for parsing. In: Proc. NAACL 2006 (2006)
17. Peng, F., Feng, F., McCallum, A.: Chinese segmentation and new word detection using conditional random fields. In: Proc. of COLING 2004 (2004)
18. Sproat, R., Gale, W., Shih, C., Chang, N.: A stochastic finite-State word-segmentation algorithm for Chinese. Computational Linguistics 22(3) (1996)
19. Tseng, H., Chang, P., Andrew, G., Jurafsky, D., Manning, C.: A conditional random field word segmenter for SIGHAN Bakeoff 2005. In: Proc. SIGHAN Workshop (2005)
20. Xue, N., Shen, S.: Chinese word segmentation as LMR tagging. In: Proceedings of the Second SIGHAN Workshop, pp. 176–179 (2003)