# Word-Based and Character-Based Word Segmentation Models: Comparison and Combination

**Weiwei Sun**
Department of Computational Linguistics, Saarland University
German Research Center for Artificial Intelligence (DFKI)
wsun@coli.uni-saarland.de

## Abstract

We present a theoretical and empirical comparative analysis of the two dominant categories of approaches in Chinese word segmentation: word-based models and character-based models. We show that, in spite of similar performance overall, the two models produce different distribution of segmentation errors, in a way that can be explained by theoretical properties of the two models. The analysis is further exploited to improve segmentation accuracy by integrating a word-based segmenter and a character-based segmenter. A *Bootstrap Aggregating* model is proposed. By letting multiple segmenters vote, our model improves segmentation consistently on the four different data sets from the second SIGHAN bakeoff.

## 1 Introduction

To find the basic language units, i.e. words, segmentation is a necessary initial step for Chinese language processing. There are two dominant models for Chinese word segmentation. The first one is what we call "word-based" approach, where the basic predicting units are words themselves. This kind of segmenters sequentially decides whether the local sequence of characters make up a word. This word-by-word approach ranges from naive maximum matching (Chen and Liu, 1992) to complex solution based on semi-Markov conditional random fields (CRF) (Andrew, 2006). The second is "character-based" approach, where basic processing units are characters which compose words. Segmentation is formulated as a classification problem to predict whether a character locates at the beginning of, inside or at the end of a word. This character-by-character method was first proposed in (Xue, 2003), and a number of sequence labeling algorithms have been exploited.

This paper is concerned with the behavior of different segmentation models in general. We present a theoretical and empirical comparative analysis of the two dominant approaches. Theoretically, these approaches are different. The word-based models do prediction on a dynamic sequence of possible words, while character-based models on a static character sequence. The former models have a stronger ability to represent word token features for disambiguation, while the latter models can better induce a word from its internal structure. For empirical analysis, we implement two segmenters, both using the *Passive-Aggressive* algorithm (Crammer et al., 2006) to estimate parameters. Our experiments indicate that despite similar performance in terms of overall F-score, the two models produce different types of errors, in a way that can be explained by theoretical properties. We will present a detailed analysis that reveals important differences of the two methods in Sec. 4.

The two types of approaches exhibit different behaviors, and each segmentation model has strengths and weaknesses. We further consider integrating word-based and character-based models in order to exploit their complementary strengths and thereby improve segmentation accuracy beyond what is possible by either model in isolation. We present a *Bootstrap Aggregating* model to combine multiple segmentation systems. By



*Coling 2010: Poster Volume*, pages 1211–1219,
Beijing, August 2010

letting multiple segmenters vote, our combination model improves accuracy consistently on all the four different segmentation data sets from the second SIGHAN bakeoff. We also compare our integrating system to the state-of-the-art segmentation systems. Our system obtains the highest reported F-scores on three data sets.

## 2 Two Methods for Word Segmentation

First of all, we distinguish two kinds of "words": (1) Words in dictionary are word types; (2) Words in sentences are word tokens. The goal of word segmentation is to identify word tokens in a running text, where a large dictionary (i.e. list of word types) and annotated corpora may be available. From the view of *token*, we divide segmentation models into two main categories: word-based models and character-based models. There are two key points of a segmentation model: (1) How to decide whether a local sequence of characters is a word? (2) How to do disambiguation if ambiguous segmentation occurs? For each model, we separately discuss the strategies for word prediction and segmentation disambiguation.

### 2.1 Word-Based Approach

It may be the most natural idea for segmentation to find word tokens one by one. This kind of segmenters read the input sentences from left to right, predict whether current piece of continuous characters is a word token. After one word is found, segmenters move on and search for next possible word. There are different strategies for the word prediction and disambiguation problems. Take for example maximum matching, which was a popular algorithm at the early stage of research (Chen and Liu, 1992). For word prediction, if a sequence of characters appears in a dictionary, it is taken as a word candidate. For segmentation disambiguation, if more than one word types are matched, the algorithm chooses the longest one.

In the last several years, machine learning techniques are employed to improve word-based segmentation, where the above two problems are solved in a uniform model. Given a sequence of characters $\mathbf{c} \in \mathcal{C}^n$ ($n$ is the number of characters), denote a segmented sequence of words $\mathbf{w} \in \mathcal{W}^m$ ($m$ is the number of words, i.e. $m$ varies with $\mathbf{w}$), and a function GEN that enumerates a set of segmentation candidates $\mathrm{GEN}(\mathbf{c})$ for $\mathbf{c}$. In general, a segmenter solves the following "argmax" problem:

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w} \in \mathrm{GEN}(\mathbf{c})} \theta^\top \Phi(\mathbf{c}, \mathbf{w}) \quad (1)$$

$$= \arg \max_{\mathbf{w} \in \mathrm{GEN}(\mathbf{c})} \theta^\top \sum_{i=1}^{|\mathbf{w}|} \phi(\mathbf{c}, w_{[1:i]}) \quad (2)$$

where $\Phi$ and $\phi$ are global and local feature maps and $\theta$ is the parameter vector to learn. The inner product $\theta^\top \phi(\mathbf{c}, w_{[1:i]})$ can been seen as the confidence score of whether $w_i$ is a word. The disambiguation takes into account confidence score of each word, by using the sum of local scores as its criteria. Markov assumption is necessary for computation, so $\phi$ is usually defined on a limited history. Perceptron and semi-Markov CRFs were used to estimate $\theta$ in previous work (Zhang and Clark, 2007; Andrew, 2006).

### 2.2 Character-Based Approach

Most previous data-driven segmentation solutions took an alternative, character-based view. This approach observes that by classifying characters as different positions in words, segmentation can be treated as a sequence labeling problem, assigning labels to the characters in a sentence indicating whether a character $c_i$ is a single character word (*S*) or the begin (*B*), middle (*I*) or end (*E*) of a multi-character word. For word prediction, word tokens are inferred based on the character classes. The main difficulty of this model is character ambiguity that most Chinese characters can occur in different positions within different words. Linear models are also popular for character disambiguation (i.e. segmentation disambiguation). Denote a sequence of character labels $\mathbf{y} \in \mathcal{Y}^n$, a linear model is defined as:

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \mathcal{Y}^{|\mathbf{c}|}} \theta^\top \Psi(\mathbf{c}, \mathbf{y}) \quad (3)$$

$$= \arg \max_{\mathbf{y} \in \mathcal{Y}^{|\mathbf{c}|}} \theta^\top \sum_{i=1}^{|\mathbf{c}|} \psi(\mathbf{c}, y_{[1:i]}) \quad (4)$$

Note that local feature map $\psi$ is defined only on the sequence of characters and their labels.

Several discriminative models have been exploited for parameter estimation, including perceptron, CRFs, and discriminative latent variable CRFs (Jiang et al., 2009; Tseng, 2005; Sun et al., 2009b).

### 2.3 Theoretical Comparison

Theoretically, the two types of models are different. We compare them from four aspects.

#### 2.3.1 Internal Structure of Words

Chinese words have internal structures. In most cases, Chinese character is a morpheme which is the smallest meaningful unit of the language. Though we cannot exactly infer the meaning of a word from its character components, the character structure is still meaningful. Partially characterizing the internal structures of words, one advantage of character-based models is the ability to induce new words. E.g., character "者/person" is usually used as a suffix meaning "one kind of people". If a segmenter never sees "工作者/worker" in training data, it may still rightly recognize this word by analyzing the prefix "工作/work" with label *BI* and the suffix "者" with label *E*. In contrast, current word-based models only utilize the weighted features as word prediction criteria, and thus word formation information is not well explored. For more details about Chinese word fomation, see (Sun et al., 2009a).

#### 2.3.2 Linearity and Nonlinearity

A majority of structured prediction models are linear models in the sense that the score functions are linear combination of parameters. Both previous solutions for word-based and character-based systems utilize linear models. However, both "linear" models incur nonlinearity to some extent. In general, a sequence classification itself involves nonlinearity in a way that the features of current token usually encode previous state information which is linear combination of features of previous tokens. The interested readers may consult (Liang et al., 2008) for preliminary discussion about the nonlinearity in structured models. This kind of nonlinearity exists in both word-based and character-based models. In addition, in most character-based models, a word should take a *S* label or start with a *B* label, end with *E* label, and only have *I* label inside. This inductive way for word prediction actually behaves nonlinearly.

#### 2.3.3 Dynamic Tokens or Static Tokens

Since word-based models take the sum of part score of each individual word token, it increases the upper bound of the whole score to segment more words. As a result, word-based segmenter tends to segment words into smaller pieces. A difficult case occurs when a word token $w$ consists of some word types which could be separated as words on their own. In such cases a word-based segmenter more easily splits the word into individual words. For example, in the phrase "四千三百/4300 米/meter (4300 meters)", the numeral "四千三百" consists of two individual numeral types "四千 (4000)" and "三百(300)". A word-based segmenter more easily made a mistake to segment two word tokens. This phenomenon is very common in named entities.

#### 2.3.4 Word Token or Word Type Features

In character-based models, features are usually defined by the character information in the neighboring $n$-character window. Despite a large set of valuable features that could be expressed, it is slightly less natural to encode predicted word token information. On the contrary, taking words as dynamic tokens, it is very easy to define word token features in a word-based model. Word-based segmenters hence have greater representational power. Despite of the lack of word token representation ability, character-based segmenters can use word type features by looking up a dictionary. For example, if a local sequence of characters following current token matches a word in a dictionary; these word types can be used as features. If a string matches a word type, it has a very high probability (ca. 90%) to be a word token. So word type features are good approximation of word token features.

## 3 Baseline Systems

For empirical analysis, we implement segmenters in word-based and character-based architectures respectively. We introduce them from three aspects: basic models, parameter estimation and feature selection.

**Algorithm 1**: The *PA* learning procedure.

**input** : Data $\{(\mathbf{x}_t, \mathbf{y}_t), t = 1, 2, ..., n\}$

1 Initialize: $\mathbf{w} \leftarrow (0, ..., 0)$
2 **for** $I = 1, 2, ...$ **do**
3  **for** $t = 1, ..., n$ **do**
4  Predict: $\mathbf{y}_t^* = \arg\max_{\mathbf{y} \in \text{GEN}(\mathbf{x}_t)} \mathbf{w}^\top \Phi(\mathbf{x}_t, \mathbf{y})$
5  Suffer loss: $l_t = \rho(\mathbf{y}_t, \mathbf{y}_t^*) + \mathbf{w}^\top \Phi(\mathbf{x}_t, \mathbf{y}_t^*) - \mathbf{w}^\top \Phi(\mathbf{x}_t, \mathbf{y}_t)$
6  Set: $\tau_t = \frac{l_t}{||\Phi(\mathbf{x}_t, \mathbf{y}_t^*) - \Phi(\mathbf{x}_t, \mathbf{y}_t)||^2 + 0.5C}$
7  Update: $\mathbf{w} \leftarrow \mathbf{w} + \tau_t(\Phi(\mathbf{x}_t, \mathbf{y}_t) - \Phi(\mathbf{x}_t, \mathbf{y}_t^*))$
8  **end**
9 **end**

### 3.1 Models

For both word-based and character-based segmenters, we use linear models introduced in the section above. We use a first order Markov models for training and testing. In particular, for word-based segmenter, the local feature map $\phi(\mathbf{c}, w_{[1:i]})$ is defined only on $\mathbf{c}, w_{i-1}$ and $w_i$, and thereby Eq. 2 is defined as $\hat{\mathbf{w}} = \arg\max_{\mathbf{w} \in \text{GEN}(\mathbf{c})} \theta^\top \sum_{i=1}^{|\mathbf{w}|} \phi(\mathbf{c}, w_{i-1}, w_i)$. This model has a first-order Semi-Markov structure. For decoding, Zhang and Clark (2007) used a beam search algorithm to get approximate solutions, and Sarawagi and Cohen (2004) introduced a Viterbi style algorithm for exact inference. Since the exact inference algorithm is efficient enough, we use this algorithm in our segmenter at both training and testing time.

For our character-based segmenter, the local feature map $\psi(\mathbf{c}, y_{[1:i]})$ is defined on $\mathbf{c}, y_{i-1}$ and $y_i$, and Eq. 4 is defined as $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}^{|\mathbf{c}|}} \theta^\top \sum_{i=1}^{|\mathbf{c}|} \psi(\theta, y_{i-1}, y_i)$. In our character-based segmenter, we also use a Viterbi algorithm for decoding.

### 3.2 Learning

We adopt *Passive-Aggressive* (PA) framework (Crammer et al., 2006), a family of margin based online learning algorithms, for the parameter estimation. It is fast and easy to implement. Alg. 1 illustrates the learning procedure. The parameter vector $\mathbf{w}$ is initialized to $(0, ..., 0)$. A *PA* learner processes all the instances ($t$ is from 1 to $n$) in each iteration ($I$). If current hypothesis ($\mathbf{w}$) fails to predict $\mathbf{x}_t$, the learner update $\mathbf{w}$ through calculating the loss $l_t$ and the difference between $\Phi(\mathbf{x}_t, \mathbf{y}_t^*)$ and $\Phi(\mathbf{x}_t, \mathbf{y}_t)$ (line 5-7). There are three variants in the update step. We here only present the PA-II rule[1], which performs best in our experiments.

The PA algorithm utilizes a paradigm of cost-sensitive learning to resolve structured prediction. A cost function $\rho$ is necessary to calculate the loss $l_t$ (line 5). For every pair of labels $(\mathbf{y}^*, \mathbf{y})$, users should define a cost $\rho(\mathbf{y}^*, \mathbf{y})$ associated with predicting $\mathbf{y}^*$ when the correct label is $\mathbf{y}$. $\rho$ should be defined differently for different purposes. There are two natural costs for segmentation: (1) sum of the number of wrong and missed word predictions and (2) sum of the number of wrongly classified characters. We tried both cost functions for both models. We find that the first one is suitable for word-based segmenter and the second one is suitable for character-based segmenter. We do not report segmentation performance with "weaker" cost in later sections. $C$ (in line 6) is the slack variable. In our experiments, the segmentation performance is not sensitive to $C$. In the following experiments, we set $C = 1$.

### 3.3 Features

#### 3.3.1 Word-based Segmenter

For the convenience of illustration, we denote a candidate word token $w_i$ with a context $c_{j-1}[w_{i-1}c_j...c_k][w_i c_{k+1}...c_l]c_{l+1}$.

The character features includes,

*Boundary character unigram*: $c_j$, $c_k$, $c_{k+1}$, $c_l$ and $c_{l+1}$; *Boundary character bigram*: $c_k c_{k+1}$ and $c_l c_{l+1}$.

*Inside character unigram*: $c_s$ $(k+1 < s < l)$; *Inside character bigram*: $c_s c_{s+1}$ $(k+1 < s < l)$.

*Length of current word*.

Whether $c_{k+1}$ and $c_{k+1}$ are *identical*.

*Combination Features*: $c_{k+1}$ and $c_l$,

The word token features includes,

*Word Unigram*: previous word $w_{i-1}$ and current word $w_i$; *Word Bigram*: $w_{i-1}w_i$.

[1] See the original paper for more details.

The *identity* of $w_i$, if it is a *Single character word*.

*Combination Features*: $w_{i-1}$ and length of $w_i$, $w_i$ and length of $w_{i-1}$. $c_{k+1}$ and length of $w_i$, $c_l$ and length of $w_i$.

#### 3.3.2 Character-based Segmenter

We use the exact same feature templates discribed in (Sun et al., 2009b). The features are divided into two types: character features and word type features. Note that the word type features are indicator functions that fire when the local character sequence matches a word unigram or bigram. Dictionaries containing word unigrams and bigrams was collected from the training data. Limited to the document length, we do not give the discription for the features. We suggest readers to refer to the original paper for details.

## 4 Empirical Analysis

We present a series of experiments that relate segmentation performance to a set of properties of input words. We argue that the results can be correlated to specific theoretical aspects of each model.

### 4.1 Experimental Setting

We used the data provided by the second SIGHAN Bakeoff (Emerson, 2005) to test the two segmentation models. The data contains four corpora from different sources: Academia Sinica Corpus (AS), City University of Hong Kong (CU), Microsoft Research Asia (MSR), and Peking University (PKU). There is no fixed standard for Chinese word segmentation. The four data sets above are annotated with different standards. To catch general properties, we do experiments on all the four data sets. Three metrics were used for evaluation: precision (P), recall (R) and balanced F-score (F) defined by 2PR/(P+R).

### 4.2 Baseline Performance

Tab. 1 shows the performance of our two segmenters. Numbers of iterations are respectively set to 15 and 20 for our word-based segmenter and character-based segmenter. The word-based segmenter performs slightly worse than the character-based segmenter. This is different from the experiments reported in (Zhang and Clark, 2007). We think the main reason is that we use a different learning architecture.

| | Model | P(%) | R(%) | F |
|---|---|---|---|---|
| AS | Character | 94.8 | 94.7 | 94.7 |
| | Word | 93.5 | 94.8 | 94.2 |
| CU | Character | 95.5 | 94.6 | 95.0 |
| | Word | 94.4 | 94.7 | 94.6 |
| MSR | Character | 96.1 | 96.5 | 96.3 |
| | Word | 96.0 | 96.3 | 96.1 |
| PKU | Character | 94.6 | 94.9 | 94.8 |
| | Word | 94.7 | 94.3 | 94.5 |

Table 1: Baseline performance.

### 4.3 Word Frequency Factors

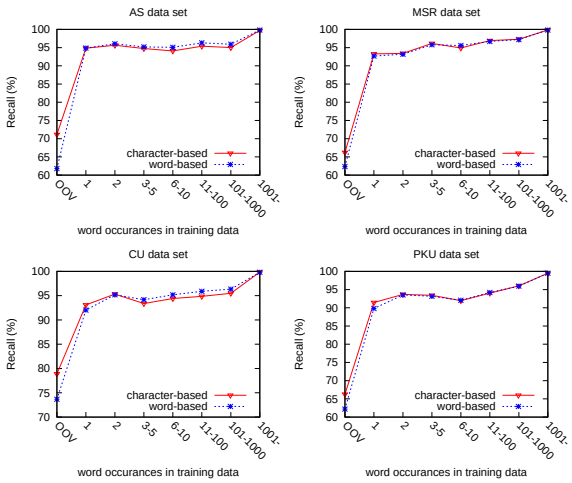


Figure 1: Segmentation recall relative to gold word frequency.

Our theoretical analysis also suggests that character-based has stronger word induction ability because it focuses more on word internal structures and thereby expresses more nonlinearity. To test the word induction ability, we present the recall relative to word frequency. If a word appears in a training data many times, the learner usually works in a "memorizing" way. On the contrary, infrequent words should be correctly recognized in a somehow "inductive" way. Fig. 1 shows the recall change relative to word frequency in each training data. Note that, the words with frequency 0 are out-of-vocabulary (OOV) words. We can clearly see that character-based model outperforms word-based model for infrequent word, especially OOV words, recognition. The "memoriz-

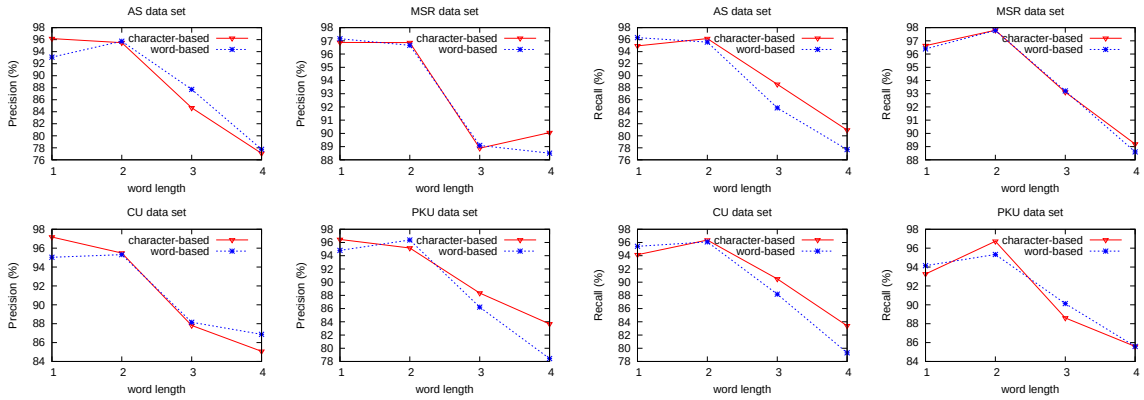


Figure 2: Segmentation precision/recall relative to gold word length in training data.

ing" ability of the two models is similar; on the AS and CU data sets, the word-based model performs slightly better. Neither model is robust enough to reliably segment unfamiliar words. The recall of OOV words is much lower than in-vocabulary words.

### 4.4 Length Factors

| Length | AS | CU | MSR | PKU |
|---|---|---|---|---|
| 1 | 61254 | 19116 | 48092 | 45911 |
| 2 | 52268 | 18186 | 49472 | 49861 |
| 3 | 6990 | 2682 | 4652 | 5132 |
| 4 | 1417 | 759 | 2711 | 2059 |
| 5(+) | 690 | 193 | 1946 | 656 |

Table 2: Word length statistics on test sets.

Tab. 2 shows the statistics of word counts relative to word length on each test data sets. There are much less words with length more than 4. Analysis on long words may not be statistical significant, so we only present length factors on small words (length is less than 5). Fig. 2 shows the precision/recall of both segmentation models relative sentence length. We can see that word-based model tends to predict more single character words, but making more mistakes. Since about 50% word tokens are single-character words, this is one main source of error for word-segmenter. This can be explained by theoretical properties of dynamic token prediction discussed in Sec. 2.3.3. The score of a word boundary assignment in a word-based segmenter is defined like $\theta^\top \sum_{i=1}^{|\mathbf{w}|} \phi(\mathbf{c}, w_{[1:i]})$. The upper bound of this score varies with the length $|\mathbf{w}|$. If a segmentation result is with more fragments, i.e. $|\mathbf{w}|$ is larger, the upper bound of its score is higher. As a result, in many cases, a word-based segmenter prefers shorter words, which may cause errors.

### 4.5 Feature Factors

We would like to measure the effect of features empirically. In particular, we do not use dynamic *word token features* in our word-based segmenter, and *word type features* in our character-based segmenter as comparison with "standard" segmenters. The difference in performance can be seen as the contribution of *word features*. There are obvious drops in both cases. Though it is not a fair comparison, word token features seem more important, since the numerical decrease in the word-based experiment is larger.

| | word-based | | character-based | |
|---|---|---|---|---|
| | $-$ | $+$ | $-$ | $+$ |
| AS | 93.1 | 94.2 | 94.1 | 94.7 |
| CU | 92.6 | 94.6 | 94.2 | 95.0 |
| MSR | 95.7 | 96.1 | 95.8 | 96.3 |
| PKU | 93.3 | 94.5 | 94.4 | 94.8 |

Table 3: F-score of two segmenters, with $(-)$ and without $(+)$ *word token/type features*.

### 4.6 Discussion

The experiments highlight the fundamental difference between word-based and character-based models, which enlighten us to design new models. The above analysis indicates that the theoretical differences cause different error distribution.

The two approaches are either based on a particular view of segmentation. Our analysis points out several drawbacks of each one. It may be helpful for both models to overcome their shortcomings. For example, one weakness of word-based model is its word induction ability which is partially caused by its neglect of internal structure of words. A word-based model may be improved by solving this problem.

## 5 System Combination

The error analysis also suggests that there is still space for improvement, just by combining the two existing models. Here, we introduce a classifier ensemble method for system combination.

### 5.1 Upper Bound of System Combination

To get an upper bound of the improvement that can be obtained by combining the strengths of each model, we have performed an oracle experiment. We think the optimal combination system should choose the right prediction when the two segmenters do not agree with each other. There is a *gold segmenter* that generates gold-standard segmentation results. In the oracle experiment, we let the three segmenters, i.e. baseline segmenters and the gold segmenter, vote. The three segmenters output three segmentation results, which are further transformed into IOB2 representation (Ramshaw and Marcus, 1995). Namely, each character has three *B* or *I* labels. We assign each character an oracle label which is chosn by at least two segmenters. When the baseline segmenters are agree with each other, the gold segmenter cannot change the segmentation whether it is right or wrong. In the situation that the two baseline segmenters disagree, the vote given by the gold segmenter will decide the right prediction. This kind of optimal performance is presented in Tab. 4. Compared these results with Tab. 1, we see a significant increase in accuracy for the four data sets. The upper bound of error reduction with system combination is over 30%.

### 5.2 Our Model

Bootstrap aggregating (Bagging) is a machine learning ensemble meta-algorithm to improve classification and regression models in terms of stability and classification accuracy (Breiman, 1996). It also reduces variance and helps to avoid overfitting. Given a training set $D$ of size $n$, Bagging generates $m$ new training sets $D_i$ of size $n' \leq n$, by sampling examples from $D$ uniformly. The $m$ models are fitted using the above $m$ bootstrap samples and combined by voting (for classification) or averaging the output (for regression).

We propose a Bagging model to combine multiple segmentation systems. In the training phase, given a training set $D$ of size $n$, our model generates $m$ new training sets $D_i$ of size $63.2\% \times n$ by sampling examples from $D$ without replacement. Namely no example will be repeated in each $D_i$. Each $D_i$ is separately used to train a word-based segmenter and a character-based segmenter. Using this strategy, we can get $2m$ *weak* segmenters. Note that the sampling strategy is different from the standard one. Our experiment shows that there is no significant difference between the two sampling strategies in terms of accuracy. However, the *non-placement* strategy is more efficient. In the segmentation phase, the $2m$ models outputs $2m$ segmentation results, which are further transformed into IOB2 representation. In other words, each character has $2m$ *B* or *I* labels. The final segmentation is the voting result of these $2m$ labels. Note that since $2m$ is an even number, there may be equal number of *B* and *I* labels. In this case, our system prefer *B* to reduce error propagation.

| | P(%) | R(%) | F | ER (%) |
|---|---|---|---|---|
| AS | 96.6 | 96.9 | 96.7 | 37.7 |
| CU | 97.4 | 97.1 | 97.3 | 46.0 |
| MSR | 97.5 | 97.7 | 97.6 | 35.1 |
| PKU | 96.8 | 96.2 | 96.5 | 32.7 |

Table 4: Upper bound for combination. The error reduction (ER) rate is a comparison between the F-score produced by the oracle combination system and the character-based system (see Tab. 1).

### 5.3 Results

Fig. 4 shows the influence of $m$ in the bagging algorithm. Because each new data set $D_i$ in bagging algorithm is generated by a random procedure, the performance of all bagging experiments are not the same. To give a more stable evaluation, we repeat 5 experiments for each $m$ and show the

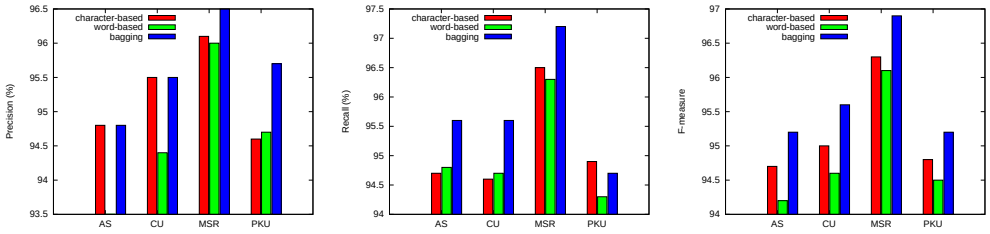


Figure 3: Precision/Recall/F-score of different models.

averaged F-score. We can see that the bagging model taking two segmentation models as basic systems consistently outperform the baseline systems and the bagging model taking either model in isolation as basic systems. An interesting phenomenon is that the bagging method can also improve word-based models. In contrast, there is no significant change in character-based models.

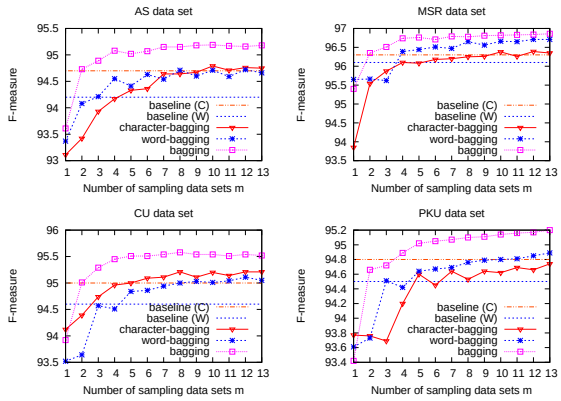


Figure 4: F-score of bagging models with different numbers of sampling data sets. *Character-bagging* means that the bagging system built on the single character-based segmenter. *Word-bagging* is named in the same way.

Fig. 3 shows the precision, recall, F-score of the two baseline systems and our final system for which we generate $m = 15$ new data sets for bagging. We can see significant improvements on the four datasets in terms of the balanced F-score. The improvement of precision and recall are not consistent. The improvement of AS and CU datasets is from the recall improvement; the improvement of PKU datasets is from the precision improvement. We think the different performance is mainly because the four datasets are annotated by using different standards.

| | AS | CU | MSR | PKU |
|---|---|---|---|---|
| (Zhang et al., 2006) | 95.1 | 95.1 | 97.1 | 95.1 |
| (Zhang and Clark, 2007) | 94.6 | 95.1 | 97.2 | 94.5 |
| (Sun et al., 2009b) | N/A | 94.6 | **97.3** | **95.2** |
| This paper | **95.2** | **95.6** | 96.9 | **95.2** |

Table 5: Segmentation performance presented in previous work and of our combination model.

Tab. 5 summarizes the performance of our final system and other systems reported in a majority of previous work. The left most column indicates the reference of previous systems that represent state-of-the-art results. The comparison of the accuracy between our integrating system and the state-of-the-art segmentation systems in the literature indicates that our combination system is competitive with the best systems, obtaining the highest reported F-scores on three data sets.

## 6 Conclusion

We have presented a thorough study of the difference between word-based and character-based segmentation approaches for Chinese. The theoretical and empirical analysis provides insights leading to better models. The strengths and weaknesses of the two methods are not exactly the same. To exploit their complementary strengths, we propose a Bagging model for system combination. Experiments show that the combination strategy is helpful.

## Acknowledgments

The work is supported by the project TAKE (Technologies for Advanced Knowledge Extraction), funded under contract 01IW08003 by the German Federal Ministry of Education and Research. The author is also funded by German Academic Exchange Service (DAAD).

## References


Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 465–472. Association for Computational Linguistics, Sydney, Australia.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Keh-Jiann Chen and Shing-Huan Liu. 1992. Word identification for mandarin Chinese sentences. In *Proceedings of the 14th conference on Computational linguistics*, pages 101–107. Association for Computational Linguistics, Morristown, NJ, USA.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:551–585.

Thomas Emerson. 2005. The second international Chinese word segmentation bakeoff. In *In Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 123–133.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and pos tagging – a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 522–530. Association for Computational Linguistics, Suntec, Singapore.

Percy Liang, Hal Daumé, III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 592–599. ACM, New York, NY, USA.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora, Cambridge, Massachusetts, USA*, pages 82–94.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *In Advances in Neural Information Processing Systems 17*, pages 1185–1192.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009a. Chinese semantic role labeling with shallow parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1475–1483. Association for Computational Linguistics, Singapore.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009b. A discriminative latent variable Chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64. Association for Computational Linguistics, Boulder, Colorado.

Huihsin Tseng. 2005. A conditional random field word segmenter. In *In Fourth SIGHAN Workshop on Chinese Language Processing*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics and Chinese Language Processing*.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for Chinese word segmentation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 193–196. Association for Computational Linguistics, New York City, USA.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847. Association for Computational Linguistics, Prague, Czech Republic.