Pruning False Unknown Words to Improve Chinese Word Segmentation

Chooi-Ling GOH and **Masayuki ASAHARA** and **Yuji MATSUMOTO**
Graduate School of Information Science
Nara Institute of Science and Technology, Japan
8916-5 Takayama, Ikoma,
Nara 630-0192,
Japan
{ling-g,masayu-a,matsu}@is.naist.jp

Abstract

During the process of unknown word detection in Chinese word segmentation, many detected word candidates are invalid. These false unknown word candidates deteriorate the overall segmentation accuracy, as it will affect the segmentation accuracy of known words. Therefore, we propose to eliminate as many invalid word candidates as possible by a pruning process. Our experiments show that by cutting down the invalid unknown word candidates, we improve the segmentation accuracy of known words and hence that of the overall segmentation accuracy.

1 Introduction

Since written Chinese texts do not use any markers such as spaces to indicate the word boundaries, word segmentation has become an essential task prior to any other Chinese language processing. There are two main problems in this task, segmentation ambiguities and unknown word occurrences. We can either solve these two problems in one single process or make them as two separate processes. Both approaches have pros and cons. If we make them as two separate processes, then we can focus on each problem independently. However, if two problems are solved in one single process, then the processing time may be shortened, but the computation will become more complicated. It is difficult to judge which approach is better.

In this paper, we would like to compare the results by using only a single model for word segmentation (Goh et al., 2004) and by joining two models (unknown word model and disambiguation model). The extracted unknown words go through a pruning step, to eliminate those unlikely to be word candidates. Our experiments show that two processes for word segmentation perform better than only with a single process.

2 Previous Work

Chinese word segmentation has been a research topic since long time ago. At first, people believed in rule based method, and tried to apply rules for word segmentation. However, as new words and new patterns can always be created, it became more and more complicated to maintain such a rule based system. Later, with the evolution of segmented corpora, it has become possible to apply machine learning based method to train a model for word segmentation.

In year 2003, a competition for Chinese word segmentation was carried out in SIGHAN[1] workshop to compare the accuracy of various methods (Sproat and Emerson, 2003). It used to be difficult to compare the accuracy of various systems because the experiments had been done on different corpora. Therefore, this bakeoff intended to standardize the training and testing corpora, so that a fair evaluation could be made. Along the history, some researches proposed solving ambiguity problem and detecting unknown word in one process (Xue and Converse, 2002; Asahara et al., 2003; Fu and Luke, 2004; Goh et al., 2004)

[1] A Special Interest Group of the Association of Computational Linguistics, http://www.sighan.org/.

and some split these processes into multiple steps (Zhang et al., 2003; Ma and Chen, 2003). Furthermore, people are combining multiple statistical models so that one model could cover the weakness of the other models. (Xue and Converse, 2002) first use a maximum entropy model to segment the text, then apply a transformational-based model to correct the output of the first model. (Asahara et al., 2003) use a hidden Markov model to segment the text for known words, then apply a support vector machine-based chunker for unknown word detection. (Fu and Luke, 2004) combine a word juncture model with word formation pattern, while (Goh et al., 2004) use maximum matching algorithm with support vector machines. From these work, we realise that by combining models of different approaches, we are able to obtain better results than only using a single model.

3 Problem Specification

The SIGHAN bakeoff results show that combining word segmentation and unknown word detection in one process produces reasonable result. Without unknown word detection, we get worse result if there are a lot of unknown words in the text. However, while the recall for unknown words increases, the recall for known words decreases. This is because those invalid detected unknown words are the cause of errors in known word segmentation. Our idea relies on the following findings. Introducing one valid unknown word creates one correct word. However introducing one invalid unknown word will possibly make (at least) two words incorrect (one unknown and one known). On the other hand, deleting one valid unknown word makes one word incorrect but deleting one invalid unknown word will possibly make two known words correct. If we can delete as many invalid words as possible, we will be able to increase the accuracy of known words and the overall segmentation.

Furthermore, the same unknown word found in one context may be missed out at another context. Therefore, after unknown word detection, we could rerun the overall segmentation again to include those missing unknown words. In short, our approach is to separate the word segmentation (disambiguation) and unknown word detection into two independent processes, so that we could focus on each problem more thoroughly and more specifically.

4 Proposed Method

Our method is based on the report by (Goh et al., 2004), where a maximum matching algorithm (MM) combining with support vector machines (SVM) model is proposed to solve the ambiguity problem and unknown word detection at the same time. In their report, if the model focuses on solving ambiguity problem, then the accuracy for known words is higher; and on the contrary if it focuses on unknown word detection, then the recall for unknown words is higher but the accuracy for known words drops. Although there is a balance point for both problems, it is quite difficult to further improve on the accuracy. Two problems are observed in (Goh et al., 2004). First, since only half of the words from the training data are used in the dictionary, some of the known words cannot be segmented correctly as they are not found in the dictionary. Second, only part of the words in the training data are used for the unknown word detection training. In other words, the training of word patterns are not thorough too. Our method intends to make full use of the training data for both problems, so that we can increase the recall for unknown words while at the same time maintains the accuracy for known words.

Figure 1 shows the flow of our process. We refer to our two models as the unknown word model and the disambiguation model. First, we use the unknown word model to extract unknown word candidates from the input text and apply a pruning process to them. Next, the new words are registered to the disambiguition model's dictionary and the final segmentation is done with the new dictionary. We will describe each step in more detail.

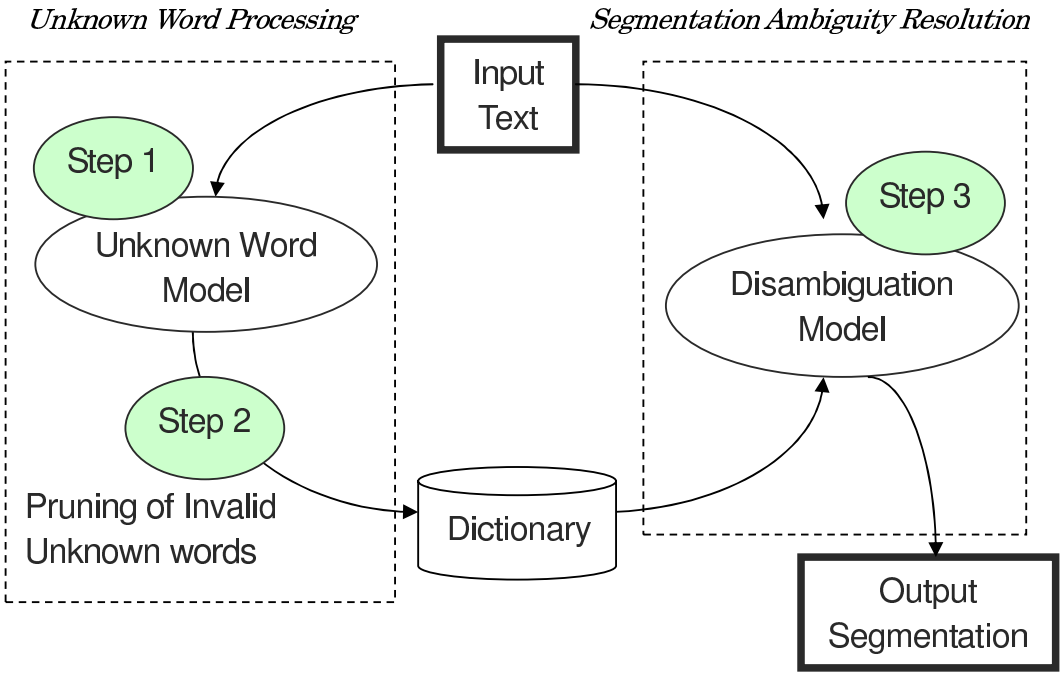

Figure 1: Segmentation Flow

4.1 Unknown Word Processing

The unknown word processing consists of two steps. First, we extract unknown word candidates with the unknown word model. Since not all extracted unknown words are valid, we then apply the second step to eliminate those invalid unknown words.

4.1.1 Unknown Word Model

In fact, the unknown word model itself is a complete word segmentation model. It could handle both disambiguation and unknown word detection in one single process. However, while the recall for unknown word increases, the accuracy for known words is affected. Since this model can get optimal result for unknown word detection, we would like to extract the unknown words in this model, meaning those words not found in the dictionary[2]. We then apply a pruning process to the unknown word candidates before registering the new words to the dictionary used in the disambiguition model for final segmentation.

The probability model used is the maximum entropy (ME) model. The ME model is similar to the one described in (Xue and Converse, 2002) with different feature templates. Lets c_i be the current character that we want to tag and i stands for the focus position. We use characters (represented by c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}), character types (represented by y_{i-2}, y_{i-1}, y_i, y_{i+1}, y_{i+2}) and previously estimated tags (represented by t_{i-2}, t_{i-1}) as the feature templates. We define four character types in our model, digits, alphabets, symbols (including punctuation marks) and hanzi (other chinese characters). The task is to estimate the tag t_i.

1. Characters. Unigram (c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}). Bigram ($c_{i-2}c_{i-1}$, $c_{i-1}c_i$, $c_{i-1}c_{i+1}$, c_ic_{i+1}, $c_{i+1}c_{i+2}$).

2. Character types. Unigram (y_{i-2}, y_{i-1}, y_i, y_{i+1}, y_{i+2}). Bigram ($y_{i-2}y_{i-1}$, $y_{i-1}y_i$, $y_{i-1}y_{i+1}$, y_iy_{i+1}, $y_{i+1}y_{i+2}$).

[2]The initial dictionary contains all words from the training data.

3. Previously estimated tags. (t_{i-2}, t_{i-1}).

We also regard the problem as a tagging problem on characters. The ME model will tag each character into one of the 4 possible tags as shown in Table 1 based on these feature templates. The B[egin], I[ntermediate], E[nd], S[ingle] tags are called LL (left boundary), MM (middle), RR (right boundary) and LR (single-character word) in (Xue and Converse, 2002).

Tag	Description
S	one-character word
B	first character in a multi-character word
I	intermediate character in a multi-character word (for words longer than two characters)
E	last character in a multi-character word

Table 1: Position tags in a word

The outputs of ME model are then converted back to word segments based on the position tags. The conversion becomes complicated when there exists inconsistency in consecutive tags. For example, it is possible that ME model assigns "SE" to two continuous characters, which is logically not allowed. Therefore, we made a slight correction to the output tags as shown in Table 2. We look at the current tag or the next tag to decide whether to make a change on previous tag or current tag. The correction does not cover all possible mistakes but only those that are seen in the outputs. The intuition behind is quite simple. We assume that when there is an "I", then is must end with an "E". Alternatively, we may trust the next coming tag, and try to change the former tag. After the correction of inconsistency tags, we convert the characters back to words. We put a word separater (a blank space) in every place that begins with either "B" or "S".

Condition	Correction
prevtag = "I" and curtag = "S"	curtag = "E"
prevtag = "B" and curtag = "S"	prevtag = "S"
prevtag = "S" and curtag = "E"	prevtag = "B"
prevtag = "S" and curtag = "I"	prevtag = "B"
prevtag = "I" and curtag = "B" and nexttag = "B"	curtag = "E"
prevtag = "B" and curtag = "B" and nexttag = "E"	prevtag = "S"
prevtag = "I" and curtag = "B" and nexttag = "S"	curtag = "E"
prevtag = "B" and curtag = "B" and nexttag = "B"	curtag = "E"
prevtag = "B" and curtag = "E" and nexttag = "E"	curtag = "I"
prevtag: previous tag, curtag: current tag, nexttag: next tag	

Table 2: Correction on output tags

From the output word segmentation, those words that are not in the dictionary will be treated as unknown word candidates, which will go through the pruning process as decribed below.

4.1.2 Pruning of Invalid Unknown Words

We apply two levels of pruning for the detected unknown word candidates. First, pruning by using adjacent words and internal components. Second, pruning by using word formation power.

The first level of pruning is by using adjacent words and internal components. Let w_{i-1}, w_i, w_{i+1} be three continuous words in the text where w_i is an unknown word candidate and $w_i = e_{i,1}e_{i,2}...e_{i,n}$ where $e_{i,j}$ is a character and n is the length of the word. We assume that if the unknown word forms a known word with adjacent characters or words, then it is not a valid unknown word. Therefore, if any one of the following words exists in the dictionary, then we delete the unknown word from the list:

1. $e_{i-1,n}e_{i,1}$ - the last character of previous word and the first character of unknown word
2. $w_{i-1}e_{i,1}$ - the previous word and the first character of unknown word
3. $e_{i-1,n}w_i$ - the last character of previous word and the unknown word
4. $w_{i-1}w_i$ - the previous word and the unknown word
5. $e_{i,n}e_{i+1,1}$ - the last character of unknown word and the first character of next word
6. $e_{i,n}w_{i+1}$ - the last character of unknown word and the next word
7. $w_ie_{i+1,1}$ - the unknown word and the first character of next word
8. w_iw_{i+1} - the unknown word and the next word

For those unknown words with length greater than 4 characters, it is possible that it includes a known word inside, especially an idiomatic phrase. Therefore, if either $e_1e_2e_3e_4$ (the first 4 characters) or $e_{n-3}e_{n-2}e_{n-1}e_n$ (the last four characters) exists in the dictionary (except those words that are numbers, alphabets or symbols), then we delete the unknown word candidate from the list.

The second level of pruning is by using word formation power (Nie et al., 1995; Fu and Luke, 2003). We define the word formation power (WFP) as below, where the $pattern$ is either S, B, I or E, introduced in Table 1.

$$pattern(e) = \frac{count(pattern(e))}{count(e)}$$

$$WFP(w) = B(e_1)\prod_{i=2}^{n-1} I(e_i)E(e_n)$$

Previous researches use a predefined threshold to eliminate the unknown words but we generate the threshold from the training corpus. The threshold is defined as the minimum WFP of words of the same length with the unknown word. Therefore, if the WFP falls in any one of the conditions below, then the unknown word candidate is deleted. However, we will accept any unknown word of one character.

1. $WFP(w)$ is less than the minimum $WFP(x)$ where $length(x) = length(w)$
2. The WFP is less than the total production of every single character in the word, $WFP(w) < S(e_1)S(e_2)...S(e_n)$
3. There exists high probability of single character in the word. Currently we run only on words where $length(w) = 4$, $WFP(w) < S(e_1)S(e_2)B(e_3)E(e_4)$ or $WFP(w) < B(e_1)E(e_2)S(e_3)S(e_4)$
4. Any one of the character in the word appears only as single character word, $S(e_j) = 1$

After the two level pruning, the unknown word candidates are registered in the dictionary for used in the disambiguation model.

4.2 Segmentation Ambiguity Resolution

We assume that there is no unknown words in the disambiguition model. If all word candidates can be found in the dictionary, we just need to solve the ambiguity problem here. Similar to (Goh et al., 2004), we use maximum matching algorithm to first segment the text forwards (FMM) and backwards (BMM), but instead of using SVM, we apply maximum entropy (ME) models for classification of characters. This is because SVM requires more computational power. Since we need to create two models, it is better if we can apply a model which can give reasonble results with lower computational power.

During the training of ME model, the dictionary used in the MM models consists of all words from the training data only. While during testing phase, the dictionary is added with the unknown words extracted from the unknown word processing phase. After the initial segmentation by using FMM and BMM models, we will convert the output words of the MMs into characters, where each character is assigned with a position tag. These tags show the character position in a word, as described in Table 1. The output of MMs will be used as features in ME models. For example, for the sentence "迎 新 春 联 谊 会 上 " (At the New Year gathering party), FMM has the position tags as "BEBESBE" and BMM has "SBEBEBE". The feature templates are as the following. Output of FMM is represented by $f_{i-2}, f_{i-1}, f_i, f_{i+1}, f_{i+2}$ and output of BMM is represented by $b_{i-2}, b_{i-1}, b_i, b_{i+1}, b_{i+2}$. Each character will be tagged by the ME model based on these features.

1. Characters. Unigram (c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}). Bigram ($c_{i-2}c_{i-1}$, $c_{i-1}c_i$, $c_{i-1}c_{i+1}$, c_ic_{i+1}, $c_{i+1}c_{i+2}$).
2. Output of FMM and BMM. ($f_{i-2}b_{i-2}$, $f_{i-1}b_{i-1}$, f_ib_i, $f_{i+1}b_{i+1}$, $f_{i+2}b_{i+2}$).
3. Previously estimated tags. (t_{i-2}, t_{i-1}).

After the character tagging, we apply the same rules for inconsistency tagging (Table 2), and finally convert the characters back to words.

5 Experiments and Results

We have run our experiments on SIGHAN Bakeoff data. There are 4 datasets provided by different instituitions. The details of the datasets are shown in Table 3. The unknown word rates vary by datasets, CHTB has the most unknown words whereas AS has the least. The size of training data is different too, AS has as big as 5.8M but HK has only 240K words. These are the main factors that affect the accuracy of the overall segmentation.

Corpus	Origin	# of train words (token)	# of test words	Unknown word rate	# of train words (type) = original dictionary
PKU	Peking University	1.1M	17,194	6.9%	55,226
CHTB	Chinese Penn Treebank	250K	39,922	18.1%	19,730
HK	Hong Kong City University	240K	34,955	7.1%	23,747
AS	Academia Sinica	5.8M	11,985	2.2%	146,226

Table 3: Bakeoff Data

5.1 Unknown Word Extraction

In Section 4.1, we have extracted the unknown words from the testing data. Table 4 shows the accuracy of the unknown word extraction. We show only the results on distinct words.

$$
\begin{aligned}
Recall &= \frac{\textit{no. of valid extracted unknown words}}{\textit{total no. of distinct unknown words in gold data}} \\
Precision &= \frac{\textit{no. of valid extracted unknown words}}{\textit{total no. of distinct unknown words extracted}} \\
\textit{F-measure} &= \frac{2 \times Recall \times Precision}{Recall + Precision}
\end{aligned}
$$

We can see from this table that after the pruning, the recalls of unknown words drop, but the precisions increase. However, the balance F-measures have increased after pruning. As we shall see in the next section, although the recalls of unknown words drop, the overall segmentation by this pruning step improves.

Corpus	Pruning	Recall	Precision	F-measure
PKU	Before	72.40	53.59	61.59
	After	66.61	61.19	63.79
CHTB	Before	69.58	49.78	58.03
	After	68.05	58.94	63.17
HK	Before	74.58	54.20	62.78
	After	69.72	61.91	65.58
AS	Before	74.85	51.41	60.95
	After	68.42	58.21	62.90

Table 4: Accuracy of Unknown Word Extraction (distinct words only)

5.2 Segmentation Result

The evaluation of word segmentation is done by using the tool provided in SIGHAN Bakeoff (Sproat and Emerson, 2003). We evaluate the performance in recall, precision and F-measure for overall segmentation, and recall for unknown words and known words.

$$
\begin{aligned}
Recall &= \frac{\textit{no. of correctly segmented words}}{\textit{total no. of words in gold data}} \\
Precision &= \frac{\textit{no. of correctly segmented words}}{\textit{total no. of words segmented}} \\
\textit{F-measure} &= \frac{2 \times Recall \times Precision}{Recall + Precision} \\
Recall_{unk} &= \frac{\textit{no. of correctly segmented unknown word}}{\textit{total no. of unknown words in gold data}} \\
Recall_{k} &= \frac{\textit{no. of correctly segmented known word}}{\textit{total no. of known words in gold data}}
\end{aligned}
$$

Figure 2 compares our results with the bakeoff results. Overall, we have out-performed almost all the participants except for CHTB dataset. In addition, our method has the highest recall for unknown words compared with others.

Table 5 shows the detail results of our system[3]. We compare the performance on with or without unknown word detection, and with or without pruning. Apparently, we need unknown word detection to

[3]Note that we have converted some ascii characters (such as numbers and alphabets) to GB or Big5 code before processing. This step will automatically make some unknown words become known words.

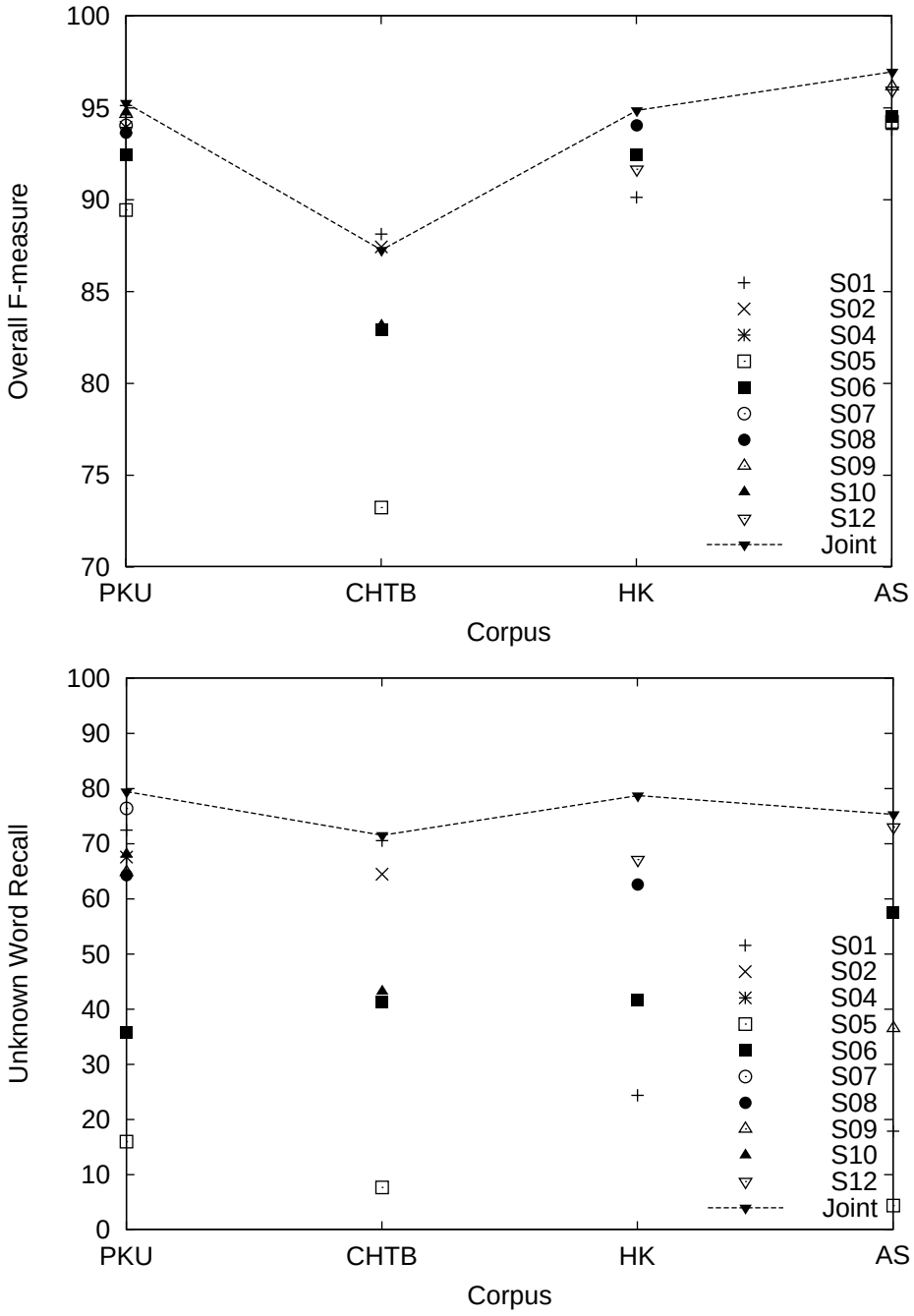

Figure 2: Compare with Bakeoff Results (Overall F-measure and Unknown Word Recall)

improve the overall segmentation. However, while the accuracy of unknown word increases (as in the row 'With unkword detection'), the accuracy of known words drops. In the next row, we have shown that re-segmentaion using the disambiguation model improves the results, as those missing words (found in one context but not the other) can be corrected. Finally, by applying the pruning step, we have again improved on the overall segmentation accuracy because some of the invalid unknown words have been eliminated. However, if the unknown word rate is low, such as AS corpus, it would be better if all the detected words are used for re-segmentation because the pruning steps eliminate too many valid unknown words (5%) relatively.

Corpus	Model	$Recall$	$Precision$	$F\text{-}measure$	$Recall_{unk}$	$Recall_k$
PKU	Disambiguition only	94.7	89.7	92.1	40.0	98.7
	With unkword detection	94.4	94.7	94.5	82.2	95.3
	Joint (without pruning)	94.4	95.4	94.9	82.8	95.3
	Joint (with pruning)	95.0	95.4	95.2	79.3	96.2
	Goh (MM+SVM)	95.5	94.1	94.7	71.0	97.3
	Nakagawa (HMM+ME)	95.7	95.2	95.4	77.4	97.0
	Peng (CRF)	94.7	93.5	94.1	66.0	n.a.
CHTB	Disambiguition only	82.2	67.4	74.1	10.5	98.0
	With unkword detection	84.5	85.4	85.0	70.0	87.7
	Joint (without pruning)	84.3	85.9	85.1	70.6	87.4
	Joint (with pruning)	86.7	87.7	87.2	71.4	90.1
	Goh (MM+SVM)	86.0	83.5	84.7	57.7	92.2
	Peng (CRF)	87.0	82.8	84.9	55.0	n.a.
HK	Disambiguition only	93.9	84.0	88.7	25.6	99.2
	With unkword detection	94.7	93.3	94.0	79.6	95.8
	Joint (without pruning)	94.7	94.2	94.4	80.6	95.8
	Joint (with pruning)	95.4	94.2	94.8	78.6	96.7
	Goh (MM+SVM)	95.4	92.1	93.7	65.5	97.7
	Nakagawa (HMM+ME)	95.1	94.8	95.0	71.5	96.9
	Peng (CRF)	94.0	91.7	92.8	53.1	n.a.
AS	Disambiguition only	97.2	94.3	95.7	23.3	98.8
	With unkword detection	97.1	96.5	96.8	79.8	97.5
	Joint (without pruning)	97.0	96.9	97.0	80.2	97.4
	Joint (with pruning)	97.2	96.7	96.9	75.2	97.6
	Goh (MM+SVM)	97.0	94.8	95.9	69.0	97.6
	Nakagawa (HMM+ME)	97.3	97.1	97.2	71.7	97.9
	Peng (CRF)	96.2	95.0	95.6	29.2	n.a.

Table 5: Segmentation Results of Joint Method

We have also compared our results with some recent works. As mentioned in the earlier section of this paper, our method is based on the report by (Goh et al., 2004). They use a combination of maximum matching algorithm and the state-of-the-art classifier, support vector machines, for segmentation. Our method has done a lot better than theirs as we can cover better the problem of known words and unknown words. The most recent works on segmentation are reported by (Nakagawa, 2004) and (Peng et al., 2004). (Nakagawa, 2004) uses word-level and character-level information for segmentation which is similar to our method. He uses a markov model for word-level probability, and maximum entropy model for character-level probability. Then he builds a lattice based on both probabilities and solves the problem by using Viterbi algorithm. Both word-level and character-level are used at the same time, and both known

word and unknown word segmentation are conducted simulteneously. His method has achieved better results than ours. The way that he uses the word-level (HMM) and character-level (ME) information in the lattice is much more efficient than our method. (Peng et al., 2004) use conditional random fields (CRF) for word segmentation. CRFs consider richer domain knowledge and are discriminatively-trained, which are often more accurate. However, in their experiment, the results shown do not out-perform our method. This could be because it is just a first trial on using CRFs for word segmentation and further survey on the feature sets is probably needed.

6 Error Analysis and Discussion

We proposed two steps of pruning in this paper. The first one is by looking at the adjacent words and the internal components. Although this method eliminates a lot of invalid unknown words but it has also eliminated some valid unknown words. For example, in the phrase "有 多 大 学 问 " (how knowledgeable), it has been segmented as "有 / 多 / 大 / 学 问 /" where "学 问 " (knowledge) is marked as an unknown word. However, since the adjacent word "大 " (big) forms a known word "大 学 " (university) with part of the unknown word, it has been deleted from the unknown word list.

The second step is by using word formation power. This method can delete those words that have very low probability to be words, but low probability does not mean invalid words most of the time. For example, the WFP of "共 兴 " (well-developed together) is 0.211, "共 贺 " (celebrate together) is 0.186 and "共 商 " (discuss together) is 0.143. Although "共 商 " has the smallest WFP, it is the only valid word amongst the three. Therefore, we still need to survey a better way for word pruning.

As we said before, the same unknown word may be found in one context but not the other. Furthermore, the same unknown word is detected differently in different context. Let's take the person name "孙 燕 姿 " (Sun yanzi) as an example. It has been segmented differently in our unknown word model. Let's consider the three phrases below.

1. "让 / 孙 燕 姿 / 不 / 同 于 / 其 他 /" (Let Sun yanzi be different from others)
2. "孙 燕 / 姿 乾 / 净 富 / 感 染 力 /" (Sun yanzi is clean and influential)
3. "自 孙 / 燕 姿 / 走 红 / 后 /" (after Sun yanzi became famous).

Our method has correctly detected the person name in the first phrase only. If we can determine that the first one is actually a correct one, then we would be able to delete those that are invalid such as in the second and third phrase. This will help to detect the words that occur frequently in the text, such as person name, place name and etc. Previous research has been done by (Shen et al., 1998) using local statistics.

7 Conclusion

In this paper, we have shown that a post-processing of unknown word detection is necessary in order to improve the accuracy of segmentation. In previous work, unknown word detection has been part of the segmentation process, but while the accuracy of unknown word increases, it has caused more errors on known words. We introduce a pruning step for eliminating invalid unknown words, so that we could increase the accuracy of known words. Although our pruning method may not be so effective in selecting valid unknown words (only around 60% precision), it has proved that by having a pruning step, it really helps in improving the overall segmentation results.

As a conclusion, detecting unknown word during the segmentation process may cause more errors on known words. A separate process for unknown word detection could help to increase the lexicon and improve on the segmentation accuracy. However, more survey and research are needed to select the correct unknown words.

References

Asahara, Masayuki, Chooi-Ling Goh, Xiaojie Wang, and Yuji Matsumoto. 2003. Combining Segmenter and Chunker for Chinese Word Segmentation. In *Proceedings of Second SIGHAN Workshop*, pages 144–147.

Fu, Guohong and K.K. Luke. 2003. An Integrated Approach for Chinese Word Segmentation. In *Proceedings of PACLIC 17*.

Fu, Guohong and K.K. Luke. 2004. Chinese Unknown Word Identification Using Class-based LM. In *Proceedings of IJCNLP*, pages 262–269.

Goh, Chooi-Ling, Masayuki Asahara, and Yuji Matsumoto. 2004. Chinese Word Segmentation by Classification of Characters. In *Proceedings of Third SIGHAN Workshop*.

Ma, Wei-Yun and Keh-Jiann Chen. 2003. A Bottom-up Merging Algorithm for Chinese Unknown Word Extraction. In *Proceedings of Second SIGHAN Workshop*, pages 31–38.

Nakagawa, Tetsuji. 2004. Chinese and Japanese Word Segmentation Using Word-Level and Character-Level Information. In *Proceedings of COLING*, pages 466–472.

Nie, Jian-Yun, Marie-Louise Hannan, and Wanying Jin. 1995. Unknown Word Detection and Segmentation of Chinese Using Statistical and Heuristic Knowledge. *Communications of COLIPS*, Vol.5:47–57.

Peng, Fuchun, Fangfang Feng, and Andrew McCallum. 2004. Chinese Segmentation and New Word Detection using Conditional Random Felds. In *Proceedings of COLING*, pages 562–568.

Shen, Dayang, Maosong Sun, and Changning Huang. 1998. The application & implementation of local statistics in Chinese unknown word identification. *Communications of COLIPS*, Vol.8.

Sproat, Richard and Thomas Emerson. 2003. The First International Chinese Word Segmentation Bakeoff. In *Proceedings of Second SIGHAN Workshop*, pages 133–143.

Xue, Nianwen and Susan P. Converse. 2002. Combining Classifiers for Chinese Word Segmentation. In *Proceedings of First SIGHAN Workshop*, pages 57–63.

Zhang, Hua-Ping, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. HHMM-based Chinese Lexical Analyzer ICTCLAS. In *Proceedings of Second SIGHAN Workshop*, pages 184–187.