An Empirical Study Of Semi-Supervised Chinese Word Segmentation Using Co-Training

Fan Yang
Nuance Communications, Inc.
fan.yang@nuance.com

Paul Vozila
Nuance Communications, Inc.
paul.vozila@nuance.com

Abstract

In this paper we report an empirical study on semi-supervised Chinese word segmentation using co-training. We utilize two segmenters: 1) a word-based segmenter leveraging a word-level language model, and 2) a character-based segmenter using character-level features within a CRF-based sequence labeler. These two segmenters are initially trained with a small amount of segmented data, and then iteratively improve each other using the large amount of unlabelled data. Our experimental results show that co-training captures 20% and 31% of the performance improvement achieved by supervised training with an order of magnitude more data for the SIGHAN Bakeoff 2005 PKU and CU corpora respectively.

1 Introduction

In the literature there exist two general models for supervised Chinese word segmentation, the word-based approach and the character-based approach. The word-based approach searches for all possible segmentations, usually created using a dictionary, for the optimal one that maximizes a certain utility. The character-based approach treats segmentation as a character sequence labeling problem, indicating whether a character is located at the boundary of a word. Typically the word-based approach uses word level features, such as word n-grams and word length; while the character-based approach uses character level information, such as character n-grams. Both approaches have their own advantages and disadvantages, and there has been some research in combining the two approaches to improve the performance of supervised word segmentation.

In this research we are trying to take advantage of the word-based and the character-based approaches in the semi-supervised setting for Chinese word segmentation, where there is only a limited amount of human-segmented data available, but there exists a relatively large amount of in-domain unsegmented data. The goal is to make use of the in-domain unsegmented data to improve the ultimate performance of word segmentation. According to Sun et al. (2009), "the two approaches [word-based and character-based approaches] are either based on a particular view of segmentation." This naturally motivates the use of co-training, which utilizes two models trained on different views of the input labeled data which then iteratively *educate* each other with the unlabelled data. At the end of the co-training iterations, the initially weak models achieve improved performance. Co-training has been successfully applied in many natural language processing tasks. In this paper we describe an empirical study of applying co-training to semi-supervised Chinese word segmentation. Our experimental results show that co-training captures 20% and 31% of the performance improvement achieved by supervised training with an order of magnitude more data for the SIGHAN Bakeoff 2005 PKU and CU corpora respectively.

In section 2 we review the two supervised approaches and co-training algorithm in more detail. In section 3 we describe our implementation of the co-training word segmentation. In section 4 we de-

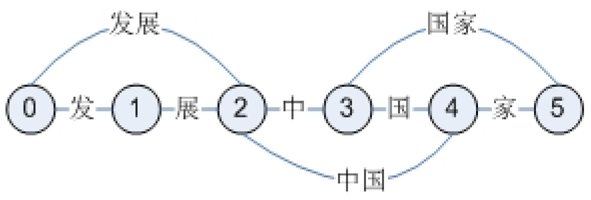

Figure 1: A search space for word segmenter

scribe our co-training experiments. In section 5 we conclude the paper.

2 Related Work

In this section, we first review the related research on the word-based and the character-based approaches for Chinese word segmentation, and comparatively analyze these two supervised approaches. We then review the related research on co-training.

2.1 Supervised Word Segmentation

2.1.1 Word-Based Segmenter

Given a character sequence $c_1c_2...c_n$, the word-based approach searches in all possible segmentations for one that maximizes a pre-defined utility function, formally represented as in Equation 1. The search space, $GEN(c_1c_2...c_n)$, can be represented as a lattice, where each vertex represents a character boundary index and each arc represents a word candidate which is the sequence of characters within the index range. A dictionary[1] can be used to generate such a lattice. For example, given the character sequence “发展中国家” and a dictionary that contains the words {发展，中国，国家} and all single Chinese characters, the search space is illustrated in Figure 1.

$$\hat{W} = arg \max_{W \in GEN(c_1c_2...c_n)} Util(W) \quad (1)$$

Dynamic programming such as Viterbi decoding is usually used to search for the optimized segmentation. The utility can be as simple as the negation of number of words (i.e. $Util(W) = - \mid W \mid$), which gives a reasonable performance if the dictionary used for generating the search space has a good coverage. Alternatively one can search for the segmentation that maximizes the word sequence probability $P(W)$ (i.e. $Util(W) = P(W)$). With a Markov assumption, $P(W)$ can be calculated using a language model as in Equation 2.

$$\begin{aligned} P(W) &= P(w_1w_2...w_m) \\ &= P(w_1).P(w_2|w_1)...P(w_n|w_1w_2...w_{n_1}) \\ &= P(w_1)P(w_2|w_1)...P(w_n|w_{n-1}) \end{aligned} \quad (2)$$

More generally, the utility can be formulated as a semi-Markov linear model, defined as Equation 3, in which Φ is the feature function vector, and Θ is the parameter vector that can be learned from training data using different techniques: Liang (2005), Gao et al. (2005), and Zhang and Clark (2007) use averaged perceptron; Nakagawa (2004) uses general iterative scaling; Andrew (2006) uses semi-Markov CRF; and Sun (2010) uses a passive-aggressive learning algorithm.

$$Util(W) \quad = \quad \Theta^T \Phi(c_1c_2...c_n, W) \quad (3)$$

2.1.2 Character-Based Segmenter

The character-based approach treats word segmentation as a character sequence labeling problem, to label each character with its location in a word, first proposed by Xue (2003).[2] The basic labeling scheme is to use two tags: ‘B’ for the beginning character of a word and ‘O’ for other characters (Peng et al., 2004). Xue (2003) use a four-tag scheme based on some linguistic intuitions: ‘B’ for the beginning character, ‘I’ for the internal characters, ‘E’ for the ending character, and ‘S’ for single-character word. For example, the word sequence “洽谈会 很 成功” can be labelled as 洽\B 谈\I 会\E 很\S 成\B 功\E. Zhao et al. (2010) further extend this scheme by using six tags.

Training and decoding of the character labeling problem is similar to part-of-speech tagging, which

[1]A dictionary is not a must to create the search space but it could shrink the search space and also lead to improved segmentation performance.

[2]Teahan et al. (2000) use a character language model to determine whether a word boundary should be inserted after each character, which can also be considered as a character-based approach as well.

is also generally formulated as a linear model. Many machine learning techniques have been explored: Xue (2003) use a maximum entropy model; Peng et al. (2004) use linear-chain CRF; Liang (2005) uses averaged perceptron; Sun et al. (2009) use a discriminative latent variable approach.

2.1.3 Comparison and Combination

It is more natural to use word-level information, such as word n-grams and word length, in a word-based segmenter; while it is more natural to use character-level information, such as character n-grams, in a character-based segmenter. Sun (2010) gives a detailed comparison of the two approaches from both the theoretical and empirical perspectives. Word-level information has greater representational power in terms of contextual dependency, while character-level information is better at morphological analysis in terms of word internal structures.

On one hand, features in a character-based model are usually defined in the neighboring n-character window; and an order-K CRF can only look at the labels of the previous K characters. Given that many words contain more than one character, a word-based model can examine a wider context. Thus the contextual dependency information encoded in a character-based model is generally weaker than in a word-based model. Andrew (2006) also shows that semi-Markov CRF makes strictly weaker independence assumptions than linear CRF and so a word-based segmenter using an order-K semi-Markov model is more expressive than a character-based model using an order-K CRF.

On the other hand, Chinese words have internal structures. Chinese characters can serve some morphological functions in a word. For example, the character 们 usually works as a suffix to signal plural; the character 者 can also be a suffix meaning a group of people; and 阿 generally works as a prefix before a person's nickname that has one character. Such morphological information is extremely useful for identifying unknown words. For example, a character-based model can learn that 阿 is usually tagged as 'B' and the next character is usually tagged as 'E'. Thus even when 阿甘 is not an existing word in the training data, a character-based model might still be able to correctly label it as 阿\B 甘\E.

Recent advanced Chinese word segmenters, either word-based or character-based, have been trying to make use of both word-level and character-level information. For example, Nakagawa (2004) integrates the search space of a character-based model into a word-based model; Andrew (2006) converts CRF-type features into semi-CRF features in his semi-Markov CRF segmenter; Sun et al. (2009) add word identify information into their character-based model; and Sun (2010) combine the two approaches at the system level using bootstrap aggregating.

2.2 Co-Training

The co-training approach was first introduced by Blum and Mitchell (1998). Theoretical analysis of its effectiveness is given in (Blum and Mitchell, 1998; Dasgupta et al., 2001; Abney, 2002). Co-training works by partitioning the feature set into two conditionally independent views (given the true output). On each view a statistical model can be trained. The presence of multiple distinct views of the data can be used to train separate models, and then each model's predictions on the unlabeled data are used to augment the training set of the other model.

Figure 2 depicts a general co-training framework. The inputs are two sets of data, a labelled set S and an unlabelled set U. Generally S is small and U is large. Two statistical models M1 and M2 are used, which are built on two sets of data L1 and L2 initialized as S but then incrementally increased in each iteration. C is a cache holding a small subset of U to be labelled by both models (Blum and Mitchell, 1998; Abney, 2002). In some applications, C is not used and both models label the whole set of U (i.e. C==U) (Collins and Singer, 1999; Nigam and Ghani, 2000; Pierce and Cardie, 2001). The stopping criteria can be, for example, when U is empty, or when a certain number of iterations are executed.

In step 5 and 6 during each iteration, some data labelled by M1 are selected and added to the training set L2, and vice versa. Several selection algorithms have been proposed. Dasgupta et al. (2001) and Abney (2002) use a selection algorithm that tries to maximize the agreement rate between the two models. The more popular selection algorithm is to choose the K examples that have the highest con-

```
Input:
    S is the labelled data
    U is the unlabelled data
Variables:
    L1 is the training data for View One
    L2 is the training data for View Two
    C is a cache holding a small subset of U
Initialization:
    L1 <- S
    L2 <- S
    C <- randomly sample a subset of U
    U <- U - C
REPEAT:
  1. Train M1 using L1
  2. Train M2 using L2
  3. Use M1 to label C
  4. Use M2 to label C
  5. Select examples labelled by M1, add to L2
  6. Select examples labelled by M2, add to L1
  7. Randomly move samples from U to C
     so that C maintains its size
UNTIL stopping criteria
```

Figure 2: A generic co-training framework

fidence score (Nigam and Ghani, 2000; Pierce and Cardie, 2001). In order to balance the class distributions in the training data L1 and L2, Blum and Mitchell (1998) select P positive examples and Q negative examples that have the highest confidence scores respectively. Wang et al. (2007) and Guz et al. (2007) use disagreement-based selection, which adds to L2, data that is labeled by M1 and M2 with high and low confidence respectively, with the intuition that such data are more useful and compensatory to M2. Finally, instead of adding the selected data to the training data, Tur (2009) propose the co-adaptation approach which linearly interpolates the existing model with the new model built with the new selected data.

3 Segmentation With Co-Training

3.1 Design of Two Segmenters

The use of co-training needs two statistical models that satisfy the following three conditions. First, in theory these two models need to be built on two conditionally independent views. However this is a very strong assumption and many large-scale NLP problems do not have a natural split of features to satisfy this assumption. In practice it has been shown that co-training can still achieve improved performance when this assumption is violated, but conforming to the conditionally independent assumption leads to a bigger gain (Nigam and Ghani, 2000; Pierce and Cardie, 2001). Thus we should strive to have the two models less correlated. Second, the two models both need to be effective for the task, that is, each of the models itself can perform the task reasonably well. Third, the decoding and training of the two models need to be efficient, as in co-training we need to segment the unlabelled data and re-train the models in each iteration. In the following we describe our design of the two segmenters.

Word-based segmenter In the word-based segmenter, we utilize a statistical n-gram language model and try to optimize the language modeling score together with a word insertion penalty, as show in Equation 4. K is a per-word penalty that is pre-determined with 10 fold cross-validation on the SIGhan PKU training set. We train a Kneser-Ney backoff language model from the training data, and extract a dictionary of words from the training data for generating the search space. Our pilot study suggested that a bigram language model is sufficient for this task.

$$Util(W) = \ln(P(W)) - |W| * K \quad (4)$$

Character-based segmenter We use an order-1 linear conditional random field to label a character sequence. Following Xue (2003), we use the four-tag scheme "BIES". We use the tool CRF++[3]. The features that we use are character n-grams within the neighboring 5-character window and tag bigrams. Given a character c_0 in the character sequence $c_{-2}c_{-1}c_0c_1c_2$, we extract the following features: character unigrams c_{-2}, c_{-1}, c_0, c_1, c_2, bigrams $c_{-1}c_0$ and c_0c_1. L2 regularization is applied in learning.

As can be seen, we build a word-based segmenter that uses only word level features, and a character-based segmenter that uses only character level features. These two segmenters by no means satisfy the conditionally independence assumption, but we have the hope that they are not too correlated as they use different levels of information and these

[3]http://crfpp.googlecode.com/svn/trunk/doc/index.html

different levels of information have been shown to be complementary in literature. Also the effectiveness of these two segmenters has been demonstrated in literature and will be shown again in our results in Section 4. Finally, both segmenters can decode and be trained pretty quickly. In our implementation, running on a Xeon 2.93GHz CPU with 4G of memory, it takes less than 30 seconds to build a word-based segmenter and less than 1 hour to build a character-based segmenter with the SIGhan PKU training data, and it takes less than 20 seconds to apply the word-based segmenter or less than 5 seconds to apply the character-based segmenter to the PKU testing data.

3.2 Co-Training

We follow the framework in Figure 2 for the co-training setup. We do not use the cache C, but directly label the whole unlabelled data set U, because in our experiment setup (see Section 4) U is not huge and computationally we can afford to label the whole set. The stopping criteria we use is when U is empty. Following Wang et al. (2007) and Guz et al. (2007), we use disagreement-based data selection. In every iteration, we pick some sentences that are segmented by the character-based model with high confidence but are segmented by the word-based model with low confidence to add to the training data of the word-based model, and vice versa. Confidence score is normalized with regard to the length of the sentence (i.e. number of characters) to avoid biasing towards short sentences. Confidence scores between the two segmenters, however, are not directly comparable. Thus we rank the sentences by their confidence scores in each segmenter respectively, and calculate the rank difference between the two segmenters. This rank difference is used as the indication of the gap of the confidence between the two segmenters. The sentences of highest rank difference are assigned to the training data of the word-based segmenter, with the segmentations from the character-based model; and the sentences of lowest rank difference are assigned to the training data of the character-based model, with segmentations from the word-based model.

4 Experiments

4.1 Data and Experiment Setup

We conduct a set of experiments to evaluate the performance of our co-training on semi-supervised Chinese word segmentation. Two corpora, the PKU corpus and the CU corpus, from the SIGhan Bakeoff 2005 are used. The PKU corpus contains texts of simplified Chinese characters, which include 19056 sentences in the training data and 1945 sentences in the testing data. The CU corpus contains texts of traditional Chinese characters, which include 53019 sentences in the training data and 1493 sentences in the testing data. The training data in each corpus is randomly split into 10 subsets. In each run one set is used as the labelled data S, and the other nine sets are combined and used as the unlabelled data U with segmentations removed. That is, 10% of the training data is used as segmented data, and 90% are used as unsegmented data in our semi-supervised training. This setup resembles our semi-supervised application, where there is only a small limited amount of segmented data but a relatively large amount of in-domain unsegmented data available. The final trained character-based and word-based segmenters from co-training are then evaluated on the testing data. Results we report in this paper are the average of the 10 runs. F-measure is used as the performance measurement. A 99% confidence interval is calculated as $\pm 2.56\sqrt{p(1-F)/N}$ for statistical significance evaluation, where F is the F-measure and N is the number of words. Subsequent assertions in this paper about statistical significance indicate whether or not the p-value in question exceeds 1%.

4.2 Co-Training Results

For comparison, we measure the baseline as the performance of a model trained with the 10% segmented data only (referred to as *BASIC* baselines). The *BASIC* baselines, both for the word-based model and the character-based model, however, use only the segmented data but leave out the large amount of available unsegmented data. We thus measure another baseline (referred to as *FOLD-IN*), which naively uses the unsegmented data. In the *FOLD-IN* baseline, a model is first trained with the 10% segmented data, and then this model is used

Table 1: Co-training results

	PKU		CU	
	char	word	char	word
BASIC	90.4	84.2	89.2	78.4
FOLD-IN	90.5	84.2	89.3	78.5
CEILING	94.5	93.0	94.2	88.9
CO-TRAINING	91.2	90.3	90.2	86.2

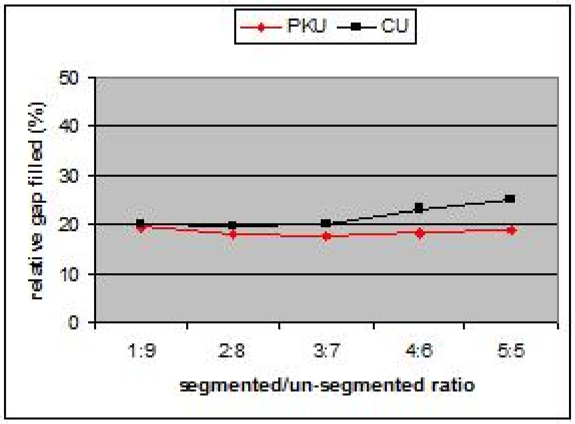

Figure 3: Gap filling with different split ratio

to label the unsegmented data. The automatic segmentation is then combined with the segmented data to build a new model. We also measure the *CEILING* as the performance of a model trained with all the training data available, i.e. we use the true segmentations of the 90% unsegmented data together with the 10% segmented data to train a model. The *CEILING* tells us the oracle performance when we have all segmented data for training, while the *BASIC* shows how much performance is dropped when we only have 10% of the segmented data. The performance of co-training will tell us how much we can fill the gap by taking advantage of the other 90% as unsegmented data in the semi-supervised training. The *FOLD-IN* baseline further verifies the effectiveness of co-training, i.e. co-training should perform better than naively folding in the unsegmented data.

Table 1 presents the results. First, we see that both the word-based and character-based models are doing a decent job under the *CEILING* condition. This confirms the effectiveness of each individual model, which is generally a requirement for running co-training. The character-based segmenter, although simple and with character-level features only, achieves the performance that is close to the state-of-the-art technologies that are much more complicated (The best performance is 95.2% for the PKU corpus and 95.1% for the CU corpus, see (Sun et al., 2009)). Second, we see that under all four conditions, the character-based segmenter performs better than the word-based model. This is not too surprising as these results are consistent with those reported in the literature. The word-based segmenter implemented in this work is less powerful, and it needs a good dictionary to achieve good performance. In our implementation, a dictionary is extracted from the segmented training set. Thus the word-based model suffers a lot when the training data is small. Third, we see that both the word-based model and the character-based model are improved by co-training, and the improvements are all statistically significant. It is not surprising for the word-based model to learn from the more accurate character-based model, which can also identify new words to add to the dictionary. More interestingly, the character-based segmenter is able to benefit from the less powerful word-based segmenter. For the character-based model, about 20% of the gap between *BASIC* and *CEILING* is filled by co-training, consistently in both the PKU and CU corpora. Finally, comparing *FOLD-IN* and *BASIC*, we see that naively using the unsegmented data does not lead to a significant improvement. This suggests that co-training provides a process that effectively makes use of the unsegmented data.

For completeness, in Figure 3 we also show the relative gap filling with different splits of the segmented vs unsegmented data. With more data moving to the segmented set, the absolute improvement of co-training over *BASIC* gets smaller, while the gap between the *BASIC* and *CEILING* also becomes smaller. The relative gap filled, i.e. the improvement relative to the difference between *BASIC* and *CEILING*, as can be seen, consistently falls inside the section of 15% and 25%.

4.3 Further Analysis

It is not surprising that the word-based segmenter benefits from co-training since it learns from the more accurate character-based segmenter. Our focus, however, is to better understand what benefit the character-based segmenter gains from the co-training procedure. The character-based segmenter treats word segmentation as a character sequence labelling problem with four tags "B I E S". Assuming that segmentation accuracy is proportional to tag accuracy, we examine the tag accuracy of the character-based segmenter before and after co-training.

If a character is labelled with tag *T0* initially before co-training and with tag *T1* after co-training, with the tag *T1* different from *T0*, there can be one of three cases: 1) *T0* is correct; 2) *T1* is correct; or 3) neither is correct. The absolute gain from co-training of switching from tag *T0* to *T1* is defined as the number of case 2 instances less case 1 instances. Absolute gain indicates the gain of tag accuracy where co-training learns to switch from *T0* to *T1*, and it contributes to the overall tag accuracy improvement. We also define *relative gain* of switching from tag *T0* to *T1* as the absolute gain divided by the total number of cases switching from tag *T0* to *T1*. Relative gain indicates how well co-training learns to switch from *T0* to *T1*.

Results are shown in Table 2. For both absolute gain and relative gain, 12 ordered switching pairs can be divided into two pools, a positive pool that has higher gain including $B \rightarrow E$, $E \rightarrow B$, $S \rightarrow B$, $S \rightarrow E$, $E \rightarrow I$, $B \rightarrow I$, $B \rightarrow S$, and a neutral pool that has lower or even negative gain including $I \rightarrow E$, $I \rightarrow S$, $I \rightarrow B$, $E \rightarrow S$, $S \rightarrow I$. The $S \rightarrow B$, $S \rightarrow E$, $B \rightarrow I$, $E \rightarrow I$ in the positive pool actually suggest that the character-based segmenter learns from co-training to combine a single-character word with it's neighbour to create a new longer word; whereas the $I \rightarrow E$, $I \rightarrow S$, $I \rightarrow B$ in the neutral pool suggest that it does not really learn how to separate a longer words into smaller units.

Table 2: Absolute Gain and Relative Gain

		Absolute Gain		Relative Gain	
T0	T1	PKU	CU	PKU	CU
B	I	678	681	0.28	0.59
B	E	2331	1727	0.41	0.46
B	S	1025	686	0.07	0.08
I	B	458	-283	0.07	-0.08
I	E	61	-1117	0.01	-0.23
I	S	323	-338	0.09	-0.34
E	B	2163	1601	0.41	0.46
E	I	963	819	0.36	0.62
E	S	520	-13	0.03	0.00
S	B	1847	892	0.27	0.30
S	I	104	47	0.22	0.28
S	E	1438	846	0.26	0.55

4.4 Feature Combination

We split the features into two sets, a character-level feature set used by the character-based segmenter and a word-level feature set used by the word-based segmenter. We have shown that these two segmenters improve each other via co-training. However, as reviewed in Section 2.1, there is active research in combining the character-level and word-level features in a segmenter. When training with the whole set of data (i.e. under the *CEILING* condition), a segmenter with combined features tends to perform better than only using one set of features. Thus we need to address two problems. First, we want to understand whether co-training, which splits the features, can actually beat the *BASIC* and *FOLD-IN* baselines of a segmenter with combined features. Second, we want to explore whether we can further improve the final co-training performance by feature combination.

To address these two problems, we adopt Weiwei Sun's character-based segmenter[4] in (Sun, 2010). We use this segmenter because it is publicly available and it performs well on both the PKU corpus and CU corpus. It models word segmentation as a character labelling problem, and solves it with a *passive-aggressive* optimization algorithm. It uses the same feature set as in (Sun et al., 2009), including both character-level features and word-level features. Character-level features include character uni-grams and bi-grams in the five character window, and whether the current character is the same as the next or the one after the next character. Word-

[4]Available at http://www.coli.uni-saarland.de/ wsun/ccws.tgz

Table 3: Sun-Segmenter's performance

	PKU	CU
BASIC	90.3	89.2
FOLD-IN	90.6	89.7
CEILING	94.8	95.0

Table 4: Results of feature combination

	PKU	CU
data combination	91.2	90.9
relabelling	91.2	91.0

level features include what word uni-grams or bi-grams are anchored at the current character. Word uni-grams and bi-grams are extracted from the labeled training data. For more details, please refer to (Sun et al., 2009) and (Sun, 2010). For ease of description, we will refer to Weiwei Sun's segmenter with combined features as *Sun-Segmenter*, and the character-based segmenter used in our co-training which uses character-level features as *Char-Segmenter*.

Table 3 shows the performance of the Sun-Segmenter under the three conditions: *BASIC*, *FOLD-IN*, and *CEILING*. We see that under the *CEILING* condition, the Sun-Segmenter outperforms the Char-Segmenter by 0.3% in the PKU corpus and 0.8% in the CU corpus. However, under the *BASIC* condition when there is only 10% of training data available, the Sun-Segmenter gives no gain. This probably is due to the fact that the Sun-Segmenter uses a much larger feature set and thus correspondingly a larger training set is needed to avoid under-fitting. The Sun-Segmenter has more gain when folding in the unsegmented data than the Char-Segmenter, further suggesting that the Sun-Segmenter is benefiting from the size of data. For both corpora, however, the Char-Segmenter after co-training beats the *FOLD-IN* baseline of the Sun-Segmenter by at least 0.5%, and the improvement is statistically significant. When there is only a small amount of segmented data available, using a more advanced segmenter with combined features still under-performs compared to co-training. These results justify the split of features for running co-training.

Next we would like to explore whether we could further improve the co-training performance, given that we have a more advanced segmenter using combined features. We try two approaches. In the first approach, after all the iterations of co-training, the data are split into two sets, one set for training the word-based segmenter L1 and the other set for training the character-based segmenter L2. The segmentations of these two sets of data are probably better than the segmentations under the *FOLD-IN* condition. We thus combine the two sets of data, and use the combined data to train a new model with the Sun-Segmenter. In the second approach, we use the character-based segmenter after co-training, which has an improved performance, to relabel the set of unsegmented data U, and then combine it with the segmented data set S. We then use the combined data to train a new model with the Sun-Segmenter.

Results are shown in Table 4. In the PKU corpus, we do not see a gain using either the data combination approach or the relabelling approach compared to the performance of the Char-Segmenter after co-training, probably because the Sun-Segmenter just modestly improves over the Char-Segmenter under the *CEILING* condition. However, in the CU corpus, where under the *CEILING* condition the Sun-Segmenter has a much bigger gain over the Char-Segmenter, there is 0.7% improvement by using the data combination approach and 0.8% by using the relabelling approach, and the improvement is statistically significant. Overall, using co-training with feature combination we are able to cut the gap between the *BASIC* baseline and *CEILING* of the Sun-Segmenter by 20% in the PKU corpus and 31% in the CU corpus.

5 Discussion

There has been some research on semi-supervised Chinese word segmentation. For example, Liang (2005) derive word cluster features and mutual information features from unlabelled data, and add them to supervised discriminative training; Li and Sun (2009) use punctuation as implicit annotations of a character starting a word (the character after a punctation) or ending a word (the character before a punctuation) in a large unlabelled data set to augment supervised data; Sun and Xu (2011) derive a large set of features from unlabelled data, including mutual information, accessor variety and punctuation variety to augment the character and word features derived from labelled data. These research works aim to use huge amount of unsegmented data to further improve the performance of an already well-trained supervised model.

In this paper, we assume a much limited amount of segmented data available, and try to boost up the performance by using in-domain unsegmented data. Chinese word segmentation is domain-sensitive or application sensitive. For example, a CRF segmenter trained on the SIGhan MSR training data, which achieves an F-measure of 96.5% in the MSR testing data, only has 83.8% when applied to the PKU testing data; and the same CRF segmenter trained on the PKU training data achieves 94.5% on the PKU testing data. When one starts a new application that requires word segmentation in a new domain, it is likely that there is only a very small amount of segmented data available.

We propose the approach of co-training for Chinese word segmentation for the semi-supervised setting where there is only a limited amount of human-segmented data available, but there exists a relatively large amount of in-domain unsegmented data. We split the feature set into character-level features and word-level features, and then build a character-based segmenter with character-level features and a word-based segmenter with word-level features, using the limited amount of available segmented data. These two segmenters then iteratively educate and improve each other by making use of the large amount of unsegmented data. Finally we combine the word-level and character-level features with an advanced segmenter to further improve the co-training performance. Our experiments show that using 10% data as segmented data and the other 90% data as unsegmented data, co-training reaches 20% performance improvement achieved by supervised training with all data in the SIGHAN 2005 PKU corpus and 31% in the CU corpus.

Acknowledgments

The authors thank Weiwei Sun for helping with data setup and technical consultation of the Sun-Segmenter. The authors also thank Christian Monson and Nicola Ueffing for helpful discussions.

References

Steven Abney. 2002. Bootstrapping. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 360–367.

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.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.

Sanjoy Dasgupta, Michael L. Littman, and David Mcallester. 2001. Pac generalization bounds for co-training. In *Proceedings of Advances in Neural Information Processing Systems*, pages 375–382.

Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computationl Linguistics*, 31(4):574.

Umit Guz, Sebastien Cuendet, Dilek Hakkani-Tur, and Gokhan Tur. 2007. Co-training using prosodic and lexical information for sentence segmentation. In *proceedings of INTERSPEECH*, pages 2597–2600.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for chinese word segmentation. *Comput. Linguist.*, 35:505–512, December.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, May.

Tetsuji Nakagawa. 2004. Chinese and japanese word segmentation using word-level and character-level information. In *Proceedings of the 20th international conference on Computational Linguistics*.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*.

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *In Proceedings of the 2001 Conference on*

Empirical Methods in Natural Language Processing, pages 1–9.

Weiwei Sun and Jia Xu. 2011. Enhancing chinese word segmentation using unlabeled data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979, Edinburgh, Scotland, UK., July.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. 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.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: comparison and combination. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1211–1219.

W. J. Teahan, Rodger McNab, Yingying Wen, and Ian H. Witten. 2000. A compression-based algorithm for chinese word segmentation. *Computational Linguistics*, 26(3):375–393, September.

Gokhan Tur. 2009. Co-adaptation: Adaptive co-training for semi-supervised learning. In *proceedings of ICASSP*, pages 3721–3724.

Wen Wang, Zhongqiang Huang, and Mary Harper. 2007. Semi-supervised learning for part-of-speech tagging of mandarin transcribed speech. In *In ICASSP*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, pages 29–48.

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, Prague, Czech Republic, June. Association for Computational Linguistics.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. A unified character-based tagging framework for chinese word segmentation. *ACM Transactions on Asian Language Information Processing*, 9(2):5:1–5:32, June.