Semi-Supervised Learning Integrated with Classifier Combination for Word Sense Disambiguation

Anh-Cuong Le [a,*], Akira Shimazu [a], Van-Nam Huynh [b], and Le-Minh Nguyen [a]

[a] *School of Information Science*
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan

[b] *School of Knowledge Science*
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan

Abstract

Word sense disambiguation (WSD) is the problem of determining the right sense of a polysemous word in a certain context. This paper investigates the use of unlabeled data for WSD within a framework of semi-supervised learning, in which labeled data is iteratively extended from unlabeled data. Focusing on this approach, we first explicitly identify and analyze three problems inherently occurred piecemeal in the general bootstrapping algorithm; namely the imbalance of training data, the confidence of new labeled examples, and the final classifier generation; all of which will be considered integratedly within a common framework of bootstrapping. We then propose solutions for these problems with the help of classifier combination strategies. This results in several new variants of the general bootstrapping algorithm. Experiments conducted on the English lexical samples of Senseval-2 and Senseval-3 show that the proposed solutions are effective in comparison with previous studies, and significantly improve supervised WSD.

Key words: Semi-supervised Learning; Word Sense Disambiguation; Computational Linguistics

* Corresponding author. Present address: College of Technology, Vietnam National University, Hanoi, 144 Xuan Thuy, Cau Giay, Hanoi, Vietnam

1 Introduction and Motivation

The automatic disambiguation of word senses has been an interest and concern since the 1950s. Roughly speaking, WSD involves the association of a given word in a text or discourse with a particular sense among numerous potential senses of that word. As mentioned in (Ide and Véronis, 1998), this is an "intermediate task" necessary in accomplishing most natural language processing tasks, such as message understanding and human-machine communication, machine translation, information retrieval, etc. Since its inception, many methods involving WSD have been developed in the literature (for a survey, see, e.g., (Ide and Véronis, 1998)).

So far, many supervised machine learning algorithms have been used for the task of WSD, including Naïve Bayes, decision tree, exemplar-based, support vector machines, maximum entropy models, etc. (see, for example, (Lee & Ng, 2002)). Due to the difficulty or the cost of obtaining labeled data, whilst unlabeled data is abundant and cheap to collect, recently several WSD studies have tried to utilize unlabeled data to boost the performance of supervised learning, e.g. (Mihalcea, 2004a; Zheng *et al.*, 2005; Su *et al.*, 2004; Pham *et al.*, 2005). The process of using both labeled and unlabeled data to build a classifier is called *semi-supervised learning*.

In the following, we first briefly introduce general approaches in semi-supervised learning for WSD. Then, we explicitly describe problems that may occur in the so-called *general bootstrapping algorithm*, which is used in this paper, and then provide some ideas for tackling these problems.

1.1 Semi-Supervised Learning Approaches

As semi-supervised learning requires less human effort for preparing annotated labeled data and potentially gives higher accuracy, it is of great interest both in theory and in practice. Semi-supervised learning methods use unlabeled data to either modify or re-prioritize hypotheses obtained from labeled data alone. In our opinion, the methods in semi-supervised learning can be grouped into two approaches, as follows.

In the first approach, the learners try to optimize parameters of the classification model using both labeled and unlabeled data. Miller & Uyar (1997), and Nigam *et al.* (2000) used a generative model for the classifier and used Expectation Maximization to estimate the model's parameters trained on both labeled and unlabeled data. Joachims (1999) used transductive inference for support vector machines to optimize performance on a specific test set, while Blum & Chawla (2001) constructed a graph based on the whole examples

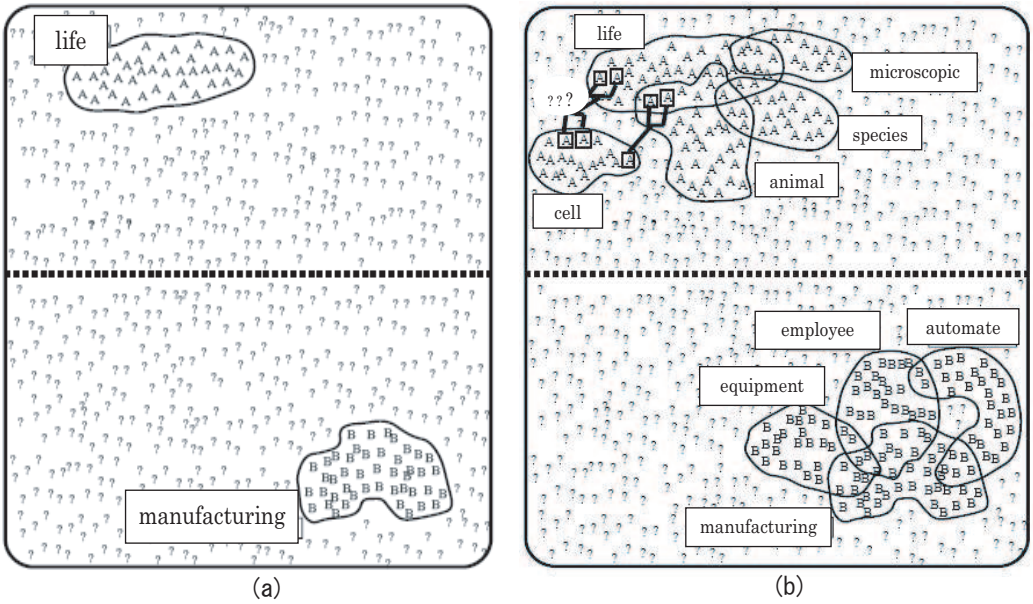

Fig. 1. A Scheme to Describe the Process of Iteratively Extending Labeled Data

and used a minimum cut on the graph to yield an optimal labeling for the unlabeled examples.

In the second approach, learners follow a strategy in which the initial labeled data is iteratively extended, and finally a larger set of labeled data is obtained and used to generate the final classifier. From the literature review, we observe that a common method for enlarging labeled data is to use the classifier trained on the current labeled dataset to detect labels for unlabeled examples. Among those new labeled examples, some highly accurate ones are selected and added to the current labeled dataset. This process is iteratively repeated until there is no unlabeled example left, or until the number of iterations reaches a predefined threshold. Two well-known methods based on this approach are self-training (Yarowsky, 1995) and co-training (Blum & Mitchell, 1998).

In this paper we follow the second approach because it seems intuitively appropriate for WSD. Fig. 1 taken from Yarowsky (1995) illustrates an example of extending labeled data in which the part (a) shows the initial contexts and part (b) shows extended labeled contexts of the polysemous word "plant". As shown in this figure, at the beginning there are some initial training contexts which contain seed collocations (i.e. features) including *life* for label "A" and *manufacturing* for label "B"(as shown in part (a)). We then use these training contexts to detect new contexts (as shown in part (b)) which contain new seed features including *animal*, *cell*, *employee*, etc. The new contexts are then used as training contexts to detect new features. This process can be repeated to recognize more new contexts, i.e. to enlarge the training dataset. Note that some new features may appear only after several extensions of training contexts, not at the first extension.

As mentioned above, we follow the second approach in which the labeled data is iteratively extended. This process is described in Algorithm 1, which can be considered as the general bootstrapping algorithm for this approach.

Algorithm 1 – The General Bootstrapping Algorithm

Input: L (labeled data); U (unlabeled data); $k = 0$; K is the maximum number of iterations

Output: H – the final classifier

1: **repeat**
2: $k \leftarrow k + 1$
3: generate classifier h trained on L
4: use h to label U, and obtain a labeled dataset U_L
5: get $L' \subset U_L$ consisting of high accuracy examples
6: $L \leftarrow L \cup L'$; $U \leftarrow U \setminus L'$
7: **until** $U = \emptyset$ or $k > K$
8: use L to generate the final classifier H

We now explicitly identify three problems (subtasks) in the general framework of semi-supervised learning which, according to our observation, may affect the performance of semi-supervised learning systems in practical applications, particularly in WSD. These problems are presented in detail below.

The first problem, denoted as P_1, regards the imbalance of labeled (training) data. We observe that if a classifier is built based on training data with a bias on certain classes (i.e., one or several classes dominate others), then this bias may become stronger at each extension of the labeled dataset. This is because a classifier tends to detect examples of dominant classes with high confidence, and consequently these examples are prioritized for a new set of labeled examples. Through steps of extending labeled data, the imbalance of labeled data is increased, which may result in decreasing the accuracy of the initial classifier. Previous studies just solved this by fixing the number of new labeled examples for each class, such as in (Blum & Mitchell, 1998; Pierce & Cardie, 2001). However, this can not be implemented in certain circumstances, for example in the case when we can not achieve enough confident new labeled examples of a class for the corresponding number which is pre-defined. To tackle this problem we will provide a procedure which can flexibly retain class distribution and avoid fixing the number of new labeled examples. In addition, the effects of using or not using this solution for this problem will be shown through experiments on Senseval-2 and Senseval-3.

The second problem, denoted by P_2, is that of how to determine a subset of new labeled examples with high confidence. It is clear that adding a large

number of misclassified examples into the labeled dataset will probably result in generating a poor classifier in the end. Therefore, one aims at obtaining new labeled examples with the highest accuracy possible. To reach this target, previous studies normally used the so-called threshold-based selection of new labeled examples. In particular, given a new example which is assigned a label with a probability of detection, a threshold value for this probability is predefined to decide whether a new labeled example will be selected or not, such as in (Yarowsky, 1995; Blum & Mitchell, 1998; Collins & Singer, 1999). However, this threshold-based method of selection may lead to a situation where choosing a higher threshold will create difficulty in extending labeled data, while it does not always result in correct classification. By contrast, a lower threshold may result in more misclassified examples, but allows more new labeled examples to be added. Therefore, the determination of a "correct" threshold in the approach becomes an important issue. In addition, determining a commonly used threshold for all unknown data is also inappropriate. To address these issues, we propose a method that flexibly and dynamically chooses an appropriate value for the threshold based on estimating the upper bound of the classification error rate of the obtained labeled dataset. Moreover, based on the observation that combining classifiers usually decrease the classification error rate, we aim at using different supervised learning algorithms to generate different classifiers and then combine them under a combination rule to increase the confidence of new labeled examples.

The third problem, denoted by P_3, is that of how to generate the final classifier when the process of extending labeled data is completed. According to the framework depicted in Algorithm 1, this process will be stopped when the number of iterations reaches a pre-specified value, or when the unlabeled dataset becomes empty. Normally, the classifier built on the labeled data obtained at the last iteration is chosen as the final one. Some studies use a development dataset to find the most appropriate value for the number of iterations, such as in (Pham *et al.*, 2005; Mihalcea, 2004a). As mentioned in problem P_2, the last classifier may be built based on new training data with some misclassified examples, so both advantages and disadvantages are concurrently brought to the last classifier. Thus, choosing the classifier trained on the last labeled dataset as the final classifier is not always be a good solution. This observation suggests that we should combine the classifiers, which are obtained at each extension of labeled data, under classifier combination strategies to utilize advantages of these different classifiers.

By reviewing various related studies, especially regarding the WSD problem, we found that most previous studies did not pay adequate attention to these three problems. There has been no study to date that could simultaneously consider these problems within a common framework of bootstrapping. In the present paper, the proposed solutions of problems mentioned above are all integrated into a common framework of bootstrapping algorithms, and as a

result a new bootstrapping algorithm is proposed. The obtained results show the effectiveness of the proposed solutions with a significant improvement of accuracy in comparison with conventional bootstrapping and supervised learning.

2 Classifier Combination

Before discussing solutions for the above-mentioned problems, it is necessary to introduce commonly used rules for combining classifiers, which will be used in the following sections.

Let $\mathcal{L} = \{l_1, \ldots, l_m\}$ be the set of labels (classes) from which each example can be assigned a label and $D = \{D_1, \ldots, D_R\}$ be a set of classifiers, each of which provides a soft decision for identifying the label of an example $\mathbf{e}$ in the form of a probability distribution over $\mathcal{L}$. Alternatively, we can define the output of i-th classifier to be a m-dimensional vector

$$D_i(\mathbf{e}) = [d_{i,1}(\mathbf{e}), \ldots, d_{i,m}(\mathbf{e})] \tag{1}$$

where $d_{i,j}(\mathbf{e})$ is the degree of "support" given by classifier D_i to the hypothesis that $\mathbf{e}$ comes from class l_j. With this notation, it is convenient to arrange the output of all R classifiers in the form of a decision matrix as follows.

$$D = \begin{bmatrix} d_{1,1}(\mathbf{e}) \ \ldots \ d_{1,j}(\mathbf{e}) \ \ldots \ d_{1,m}(\mathbf{e}) \\ \ldots \\ d_{i,1}(\mathbf{e}) \ \ldots \ d_{i,j}(\mathbf{e}) \ \ldots \ d_{i,m}(\mathbf{e}) \\ \ldots \\ d_{R,1}(\mathbf{e}) \ \ldots \ d_{R,j}(\mathbf{e}) \ \ldots \ d_{R,m}(\mathbf{e}) \end{bmatrix} \tag{2}$$

With these notation, Table 1 presents commonly used rules for combining classifiers, where $l_{\hat{j}}$ denotes the label which should be assigned to the example $\mathbf{e}$ according to the corresponding decision rule.

Also in this table, the function Δ_{ij} in majority voting is defined by:

$$\Delta_{ij} = \begin{cases} 1, & \text{if } d_{i,j}(\mathbf{e}) = \max\limits_{k} d_{i,k}(\mathbf{e}) \\ 0, & \text{otherwise} \end{cases}$$

For the details of deriving these combination rules, the reader may refer to

Table 1
Commonly Used Combination Rules

Rule	Decision Function
Max Rule	$\hat{j} = \underset{j=1\cdots m}{\arg\max} \left[\max_{i=1}^{R} d_{i,j}(\mathbf{e})\right]$
Min Rule	$\hat{j} = \underset{j=1\cdots m}{\arg\max} \left[\min_{i=1}^{R} d_{i,j}(\mathbf{e})\right]$
Median Rule	$\hat{j} = \underset{j=1\cdots m}{\arg\max} \left[\frac{1}{R}\sum_{i=1}^{R} d_{i,j}(\mathbf{e})\right]$
Majority Voting	$\hat{j} = \underset{j=1\cdots m}{\arg\max} \sum_{i=1}^{R} \Delta_{ij}$

Kittler *et al.* (1998), in which the authors presented a theoretical framework for combining classifiers using the Bayesian approach with several assumptions imposed on individual classifiers. It is worth noting that, as discussed in (Le *et al.*, 2005), these commonly used combination rules can be also generated based on the combination strategy making use of the notion of ordered weighted averaging (OWA) operators. Further, as OWA weighting vectors can be associated with linguistic quantifiers (Yager, 1988) without the strong assumptions used in Kittler *et al.* (1998), the OWA-based combination framework discussed in (Le *et al.*, 2005) allows us to interpret generated decision rules in terms of linguistic–quantifiers–based voting rules. For other decision rules derived from OWA-based combination of classifiers, the reader could refer to, e.g., (Kuncheva, 2001; Le *et al.*, 2005).

3 Proposed Solutions to Problems

Previously, we have identified three problems that may occur in semi-supervised learning methods, and observed that overcoming them may effectively enhance the performance of learning algorithms in practical situations. In this section we will discuss solutions for these problems one by one, in order to form the basic for developing a new bootstrapping algorithm in the next section.

3.1 Imbalanced Data

Now we are concerned with situations in which class distribution of the original labeled dataset is biased, i.e. one or several classes considerably dominate the others. In such a case, adding new labeled examples in semi-supervised learning may make this bias stronger. This is because following the strategy of selecting new labeled examples based on classification probability, examples of dominat classes have more chance to be selected. This is clearly an undesired situation and has been considered in previous studies, e.g. (Blum & Mitchell,

Procedure 2 – **Resize**(L_0, L', Δ): Resizing class-based subsets

Input:

L_0 is the original labeled dataset; L' is the added labeled dataset

Δ – tolerance in retaining class distribution

Output:

$N' = \{n'_1, \ldots, n'_m\}$ – new sizes of class-based subsets for L

compute $N^0 = \{n^0_1, \ldots, n^0_m\}$ – sizes of class-based subsets of L_0.
compute $N' = \{n'_1, \ldots, n'_m\}$ – sizes of class-based subsets of L'.
compute $D^0 = \{d^0_1, \ldots, d^0_m\}$ – the class distribution of L_0
repeat
  set $N = \{n_1, \ldots, n_m\}$ by $n_i = n^0 + n'_i$, for $i = 1, \ldots, m$
  compute $D = \{d_1, \ldots, d_m\}$ from N
  if there exists l such that $d_l - d^0_l > \Delta$ (*) **then**
    compute r such that $\frac{n_l - r}{\sum_{i=1}^{m} n_i - r} = (d^o_l + \Delta)$
    if $n'_l > (r + 1)$ **then**
      $n'_l \leftarrow [n'_l - (r + 1)]$
    else
      $n'_l \leftarrow 0$
    end if
  end if
until condition (*) does not hold

1998; Pierce & Cardie, 2001).

To tackle this problem, our solution is as follows. For a set of labeled examples, we divide examples into subsets such that all examples in each subset have the same label. We call these class-based subsets. From the new labeled examples which are obtained at the extension step, we must resize its class-based subsets such that the class distribution of the original labeled dataset can be retained. However, we may not always strictly retain this class distribution, such as in the case there are one or more class-based subsets which are empty. Therefore, developing a procedure for maintaining the class distribution should take this situation into account. In our case, we introduce a tolerance parameter by which we can avoid fixing the number of new labeled examples of each class is fixed, as in (Blum & Mitchell, 1998; Pierce & Cardie, 2001). The proposed solution for retaining class distribution is described in Procedure 2.

3.2 Increasing Confidence of New Labeled Data

Regarding this task, a typical approach is to use a supervised learning algorithm to train a classifier based on the labeled dataset, and then use this classifier to detect labels for the examples in a subset U' of the current unla-

beled dataset U. Formally, let $\mathcal{L}$ be the set of labels (classes), and h be the supervised classifier. Given an example $\mathbf{e}$, the classifier h applied to $\mathbf{e}$ yields a probability distribution over $\mathcal{L}$, denoted by $P_h(\cdot|\mathbf{e})$. Then it is suggested that example $\mathbf{e}$ should be assigned to label $\hat{l}$ satisfying

$$\hat{l} = \arg\max_{l \in \mathcal{L}} P_h(l|\mathbf{e})$$

If $P_h(\hat{l}|\mathbf{e})$ is greater than a threshold α, then example $\mathbf{e}$ associated with label $\hat{l}$ will be added to L.

As mentioned previously, using a classifier with threshold α for determining new labeled examples may cause a tradeoff problem between the extendibility and the accuracy of label detection. Furthermore, an increase in threshold α does not always ensure an increase in accuracy of new labeled examples. Note that the extendibility of labeled data is not only depicted by the number of new examples added, but also by the "new information" brought by these added examples. Heuristically, a new example whose label is correctly detected with low confidence may bring richer and new information to the current labeled data, and therefore it may be useful to detect labels for new examples. Therefore, it would be helpful to find out such a way of extending labeled data which can maintain the extendibility while still ensuring the accuracy of labeled data. Here we also use a threshold-based method, but instead of designing a fixed threshold, we design a set of threshold values from which the best will be chosen based on estimating the upper bound of classification error of labeled data using the approach presented in (Goldman & Zhou, 2000; Zhou & Li, 2005). Particularly, this selection of a threshold value is based on the evaluation of generated labeled datasets, which is done as follows.

As used in Goldman & Zhou (2000); Zhou & Li (2005), exploiting from (Angluin & Laird, 1988) we have a relationship between m - the size of a sequence σ and its hypothesis worst-case accuracy $(1 - \varepsilon)$ as follows:

$$m \geq \frac{2}{\varepsilon^2(1-2\eta)^2} \ln(\frac{2N}{\delta}) \tag{3}$$

where η is classification noise rate of the training data (η must be less than 0.5), N is the number of hypothesis, and δ is the confidence such that a hypothesis H that minimizes disagreement with σ will have the PAC property:

$$Pr[d(H, H^*) \geq \varepsilon] \leq \delta$$

with $d(,)$ being the sum over the probability of elements from the symmetric difference between hypothesis H and the ground-truth hypothesis H^*.

Like Zhou & Li (2005), let $c = 2\mu\ln(\frac{2N}{\delta})$ with μ making Eq. (3) hold equality, we obtain:

$$m = \frac{c}{\varepsilon^2(1-2\mu)^2}$$

or equivalently

$$\varepsilon = \sqrt{\frac{c}{m(1-2\eta)^2}} \tag{4}$$

In the Eq. (4), we can simply set c to 1, and then ε can easily be estimated through m and μ. the upper bound on the classification error rate ε will be used for evaluating new labeled dataset and from which we can determine the most appropriate α corresponding to the best new labeled dataset.

Procedure 3 – **DataEvaluate**(L_0, L_{add}): Evaluation of Generated Labeled Data

Input: L_0 is the original labeled data; L_{add} is the added labeled data

1: $m \leftarrow |L_{add}|$
2: train on L_{add} to generate classifier h
3: use h to test on L_0 and obtain classification error rate η
4: **return** $q = m(1-2\eta)^2$

Assume that we are standing at the iteration t^{th}, denote by L^t the current labeled dataset, L^t_{add} is the current added labeled dataset, and w^t is the number of examples in $L^t \cup L^t_{add}$ that are mislabeled. Then $1/\varepsilon^2$ denoted by q^t can be estimated by

$$q^t = |L^t \cup L^t_{add}| \left(1 - 2\frac{2w^t}{|L^t \cup L^t_{add}|}\right)^2$$

The semi-learning process is continued if $q^t > q^{t-1}$, and the labeled dataset is updated: $L^{t+1} \leftarrow L^t \cup L^t_{add}$.

Here we also accept this approach for evaluating the generated labeled datasets (i.e. the generated classifiers), but with a difference regarding the estimation of experimental classification error rate η. To estimate η, Goldman & Zhou (2000) used a 10-fold test on $L^t \cup L^t_{add}$, while Zhou & Li (2005) also trained on the whole data (include original labeled data and new labeled data) but just tested on the original labeled data.

Agreeing with the observation in (Zhou & Li, 2005) that it is difficult to estimate the classification error on the unlabeled examples, we therefore compute

classification error rate by testing on only the original labeled examples, with a heuristic assumption that the unlabeled examples hold the same distribution as the labeled ones. Moreover, we use only the total number of labeled examples added through iterations without the original labeled examples for training while (Goldman & Zhou, 2000) and (Zhou & Li, 2005) used both these kinds of labeled data. The reason for our choice stems from the following observations:

- WSD always has the problem of over-fitting on training data, which means if we train and test on the same data, we often receive a very high accuracy (approximate 100%).
- It is natural that if the added labeled examples are correctly classified, the classifier trained on them will give high accuracy when testing on the original labeled examples.

The flexible and dynamic strategy for selecting values for the threshold α with the help of Procedure **DataEvaluate** is used for the task of extending labeled data and this procedure is sketched in Procedure 4.

Procedure 4 – **Extendibility**$(L, U, \Omega, \mathbf{A})$: Extend labeled data

---

Input:

L – the current labeled dataset

U – the current unlabeled data

$\Omega = \{\alpha_i\}_{i=1}^{n}$ – a set of threshold values of size n

$\mathbf{A} = \{A_1, \ldots, A_R\}$ is the set of supervised algorithms, which is used in the case of using multi–classifiers

1: set a pool of empty datasets, $\mathbf{L} = \{L_i = \emptyset\}_{i=1}^{n}$
2: get classifier h trained on L
3: **for all** example $\mathbf{e} \in U$ **do**
4: use h to detect labels for $\mathbf{e}$, obtain a new labeled example $\mathbf{e}'$ with a overall corresponding support degree $P(e')$;
5: **for all** $\alpha_i \in \Omega$ **do**
6: **if** $P(e') > \alpha_i$ **then**
7: add $\mathbf{e}'$ to L_i
8: **end if**
9: **end for**
10: **end for**
11: call **Resize** on L_i and remove from L_i a certain number of examples such that it is appropriate to the new size, for $i = 1, \ldots, n$.
12: evaluate L_i by calling the procedure ***DataEvaluate***, for $i = 1, \ldots, n$, and get the best L_k, where $k = \arg\max_i DataEvaluate(L, L_i)$
13: **return** L_k

Also regarding the problem P_2, we propose a solution motivated from the observation that combining classifiers has significantly improved the performance

of supervised learning systems. Especially in the context of semi-supervised learning, where the relatively small amount of available labeled data would not be enough to build good classifiers, combination of different classifiers may hopefully enhance the quality of new labeled data by integrating complementary information extracted from individual classifiers about patterns to be classified. If we use classifier combination for detecting labels for unlabeled examples in the procedure of Extendibility, it is then called the use of multi-classifier. Otherwise, if we use only one classifier then it is called the use of single-classifier. Note that the use of either single classifier or multi-classifier is stated at Step 2 and Step 4 of Procedure 4. In the case of multi-classifier, we use a combination rule on a set of R classifiers to generate the classifier h. In such a case, R different classifiers correspond to R different machine learning algorithms taken from $\mathbf{A}$ trained on the current labeled data. For each unlabeled example $\mathbf{e}$, these R classifiers are combined to output overall support degrees associated with corresponding labels in $\mathcal{L}$. Then the highest overall support degree is used in comparison with the threshold in such a way that only the new labeled examples which have the highest overall support degrees greater than the threshold α are added to the current labeled data.

3.3 Generating the Final Classifier

Now we will discuss about problem P_3 of how to generate the final classifier. Regarding this problem, there are two issues to be addressed. The first issue is when we should stop the process of enlarging the labeled dataset, and the second one is how to build the final classifier. In previous studies related to WSD as in (Pham *et al.*, 2005; Mihalcea, 2004a), the authors first design a development dataset and then run the semi-learning algorithm on this dataset several times to select a value for the iteration number which is used for test datasets. After that the classifier built on the labeled data obtained at the last iteration is chosen as the final one.

However, in our opinion, because the optimized value of iteration number depends on each particular dataset as well as each polysemous word, it would be better if this value can be dynamically determined by evaluating generated labeled datasets. For this purpose, we can use the evaluation method described previously in Procedure 3. In addition, as discussed above, we can also use techniques of classifier combination for building the final classifier. The details are shown below.

We first observe that during the process of extending labeled data in semi-supervised learning systems for WSD, the following two situations may happen: the first one is the feature space may be also expanded concurrently, due to some new features covered by new examples which have not been occurred

in available examples previously; the second one is there may be some misclassified examples which were added to the labeled dataset at some steps of extending labeled data. Both these situations may lead to the following consequences: the generated classifiers may not much improve, or may even decrease the labelling quality for test examples which could have been correctly labeled by the initial supervised classifier; the generated classifiers may be better in detecting labels for test examples which contain many new features covered by new added labeled examples.

These observations suggest that the use of only one classifier built on the last labeled dataset as the final classifier may not always be the best solution. Again, we aim to apply strategies of combining classifiers for enhancing the labelling quality of the final classifier in semi-supervised learning for WSD. To this end, after each extension of labeled data we build the corresponding classifier and then combine all of them according to a combination strategy to obtain the final classifier. Note that, there is another alternative in which we just combine the classifiers which are trained on the original and the last labeled dataset.

4 A New Bootstrapping Algorithm

Algorithm 5 – A New Bootstrapping Algorithm

Input:L_0 is the original dataset; $\mathbf{A} = \{A_1, \ldots, A_R\}$ is the set of supervised algorithms; A^* is the primary supervised algorithm; C is a combination rule; $\mathbf{L}$ is the set of obtained labeled datasets; Δ is the tolerance which is used for retaining class distribution ; M is maximum number of unlabeled examples used at each iteration; Ω is a set of the threshold α.
Output: H – the final classifier

1: $k \leftarrow 0$; $q_0 \leftarrow 0$; $\mathbf{L} \leftarrow \{L_0\}$
2: **repeat**
3: $k \leftarrow k + 1$;
4: randomly get M examples from U to obtain $U' \subset U$
5: $L' \leftarrow$ ***Extendibility***$(L_k, U', \Omega, \mathbf{A})$
6: $L_{k+1} \leftarrow L_k \cup L'$;
7: $q_{k+1} \leftarrow$ ***DataEvaluate***(L_0, L_{k+1})
8: **if** $q_{k+1} > q_k$ **then**
9: $\mathbf{L} \leftarrow \mathbf{L} \cup \{L_{k+1}\}$;
10: **end if**
11: **until** $q_{k+1} < q_k$ or $L' = \emptyset$
12: train A^* on $\mathbf{L}$ to generate a set of classifiers, $\mathbf{h} = \{h_0, \ldots, h_t\}$
13: apply the combination rule, C, on $\mathbf{h}$ to generate the final classifier H

On the basis of the above discussions, we develop a new bootstrapping algo-

rithm as shown in Algorithm 5. In this algorithm, at each iteration firstl M examples are randomly extracted from the whole unlabeled dataset U, we denote by U' the set of these unlabeled examples (this is necessary in the case that U is very large). After U' is selected, the procedure **Extendibility** is called to enlarge the current labeled dataset L_k (k is the current iteration number). In this step, an added labeled dataset, L', is generated, which is then combined with the current labeled dataset to yield a new labeled dataset, L_{k+1}. Note that when **Extendibility** has been carried out, procedure **Resize** is called to retain class distribution for the new labeled dataset. After obtaining L_{k+1}, it is evaluated by procedure **DataEvaluate**. This process is repeated until there are no more new labeled examples to be discovered, or the new labeled examples do not improve the labeled dataset. When this process stops, we obtain a set of new labeled datasets, namely $\mathbf{L} = \{L_0, \ldots, L_t\}$ (note that, here $t = k$ or $t = k + 1$ depends on the conditions for stopping the loop of the algorithm, $q_{k+1} < q_k$ or $L' = \emptyset$). Then, we use the supervised learning algorithm A^* trained on $\mathbf{L}$ to obtain a set of different classifiers $\mathbf{h} = \{h_0, \ldots, h_t\}$. Finally, the final classifier H is generated by applying the combination rule C on $\mathbf{h}$.

Note that, at the final step of this algorithm we can also apply the combination rule for only the classifier trained on the original labeled dataset and the classifier trained on the last new labeled dataset (i.e. L_0 and L_N). This is suggested by the observation that intermediately generated classifiers participating in the combination may decrease advantages of the last classifier. Further, using only the initial and the last classifiers in combination is also due to advantages in terms of time computation and storage space.

5 Experiment

5.1 Experimental Models

Actually, the proposed semi-supervised learning algorithm is the result of integrating solutions for problems P_1, P_2, and P_3 into the general bootstrapping algorithm. Therefore, to see how effective each of the proposed solutions or their combinations is, in the sequence we develop several different experimental models of the proposed semi-supervised learning algorithm.

As in Procedure **Extendibility**, we use a set of values for α instead of a fixed value. In particular, we define this set as $\Omega = \{0.5, 0.6, 0.7, 0.8, 0.9\}$. The upper bound ($\alpha = 0.9$) and lower bound ($\alpha = 0.5$) of these values are used for those models which follow the conventional threshold-based method, which is based

Table 2
Experimental Models of Bootstrapping

Model	P_1 Solution	P_2 Solution	P_3 Solution
M_0, $\alpha = 0.9$			
M_0, $\alpha = 0.5$			
M_1, $\alpha = 0.9$	x		
M_1, $\alpha = 0.5$	x		
$M_2^{flexible+single}$	x	x (single classifier)	
$M_2^{flexible+combined}$	x	x (multiple classifiers)	
M_3^{two}	x	x	x (two classifiers)
M_3^{all}	x	x	x (all classifiers)

on a fixed threshold. Particularly, the experimental models are as follows.

- Call the general bootstrapping algorithm M_0, without any proposed solutions of P_1, P_2, and P_3. In this model, we investigate two cases: $\alpha = 0.9$ and $\alpha = 0.5$.
- To investigate problem P_1, we design the model called M_1, which is the model M_0 plus the procedure *Resize*, i.e the solution for P_1. In this model, we also investigate two cases: $\alpha = 0.9$ and $\alpha = 0.5$
- The following models are designed to test the solution of P_2 in combination with the solution of P_1, with and without using a strategy of classifier combination. Here, the set of threshold values is $\Omega = \{0.5, 0.6, 0.7, 0.8, 0.9\}$.
  - $M_2^{flexible+single}$ is the model in which we use a flexible and dynamic selection over all values in Ω for α. Moreover, this model just uses a single classifier, namely the NB classifier, to detect labels for unlabeled examples.
  - $M_2^{flexible+combined}$ is similar to $M_2^{flexible+single}$ but instead of using one classifier, here we use three classifiers including NB, MEM, and SVM, and the median rule.

  Note that all these models use procedure **DataEvaluation** as the condition for stopping the loop of the algorithm.
- Regarding problem P_3, we design two experimental models as follows: in model M_3^{two} we just combine the initial classifier and the last classifier; and in model M_3^{all} we combine initial classifier and all generated classifiers.

These models are intuitively summarized in Table 2

5.2 Data

The experiments were carried out on the datasets of Senseval-2 and Senseval-3, specifically on English lexical sample datasets. There are 73 lexical items and 57 lexical items in Senseval-2 and Senseval-3, respectively. Note that the data for each lexical item contains about 200 examples for training and 100 examples for test. For semi-supervised learning, we treat the training data as labeled data, and unlabeled examples are collected from the British National Corpus (BNC), with about 3000 examples for each lexical item. Note that the English lexical samples are also retrieved from BNC, thus for a fair test, we removed all examples from the obtained unlabeled datasets which also appear in the training or test datasets.

5.3 Features

As we know, determining the kinds of information, which are considered as evidences for disambiguating word senses, plays an important role in achieving high accuracy of WSD systems. Here we use various kinds of information as presented in (Lee & Ng, 2002), including topical context, collocations, ordered words, ordered part-of-speeches, and syntactic relations. Particularly, features used in this paper fall into the following groups:

- Topical context is represented by a set of content words that includes nouns, verbs, and adjectives, in a certain context window. Note that after these words are extracted, they will be converted to their root morphology forms for use. We designed three sets of unordered words in contexts with different windows including small, median, and large, corresponding to window sizes of 5, 10, and 50 respectively.

- A set of collocations of words. Here we design 9 collocations: $w_{-1}w_0$, w_0w_1, $w_{-2}w_{-1}w_0$, $w_{-1}w_0w_1$, $w_0w_1w_2$, $w_{-3}w_{-2}w_{-1}w_0$, $w_{-2}w_{-1}w_0$, $w_{-1}w_0w_1w_2$, and $w_0w_1w_2w_3$, where w_{-i} (w_i) is the i-th word to the left (right) of the polysemous word w_0.

- A set of words assigned with their positions in a window size of 3.

- A set of part-of-speech tags of words assigned with their positions, also in window size of 3.

- A set of syntactic features: to get syntactic information, we used shallow parsing using the chunking tool obtained from (Tsuruoka & Tsujii , 2005). From the output of this parser, we select verb-phrases and noun-phrases, and then extract from them syntactic relations including verb-noun, adjective-

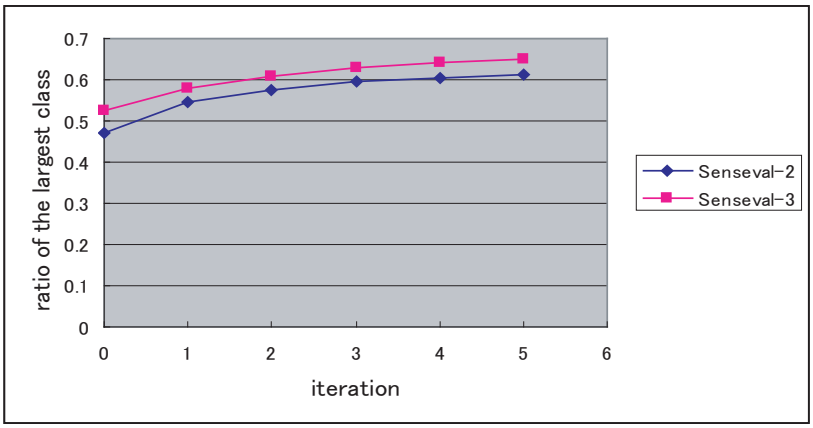

Fig. 2. Test Problem P_1 on Senseval-2 and Senseval-3

noun, and verb-adjective.

All these 7 feature sets are joined into a unique set aiming at obtaining high accuracy baselines (supervised WSD).

5.4 Supervised Algorithms and Parameter Setting

Naive Bayes (NB), MEM (Maximum Entropy Model), and SVM (Support Vector Machine) are chosen as supervised learning algorithms for procedure **Extendibility** in the case that a combination strategy is integrated in the solution of P_2. Otherwise, in the case of using single classifier instead of combining multiple classifiers we will use the NB classifier. Further, the NB algorithm is also used for A^* in Algorithm 5.

For the parameters, we set $M = 500$ and $\Delta = 0.01$.

5.5 Results

The first test is for investigating the problem of the imbalance of training data, i.e. P_1, with the experiment carried out on Senseval-2 and Senseval-3. The obtained results are shown in Fig. 2. In this experiment, we let the iterations run from 1 to 5, and compute the ratio of the largest class to the whole dataset (note that the result at iteration 0 corresponds to the original labeled data). As seen in Fig. 2, the portion of the largest class (or dominated class) increases according to the increase of iteration. This reflects the imbalance of training data is increased as discussed above.

The second test is for investigating the problem of extending labeled data

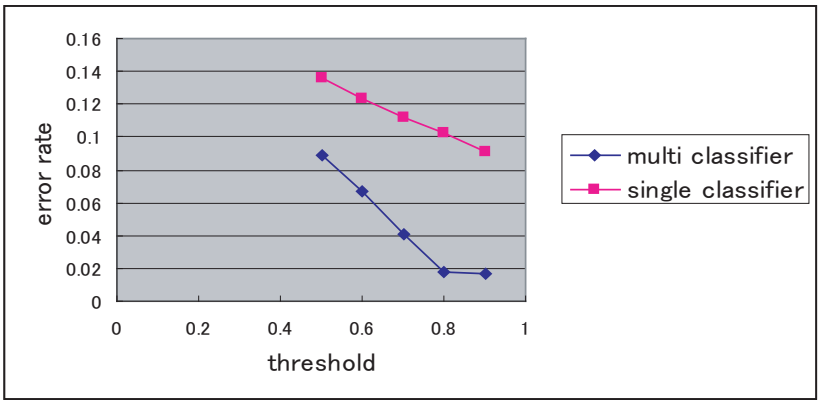

Fig. 3. Test Problem P_2 of interest, line, hard, and server datasets (show average result)

Table 3
Test Problem P_1 and P_2

	Parameter	Senseval-2	Senseval-3
Supervised Learning (NB)		64.05	71.96
M_0	$\alpha = 0.9$	62.07	71.50
M_0	$\alpha = 0.5$	62.97	71.07
M_1	$\alpha = 0.9$	63.89	71.65
M_1	$\alpha = 0.5$	64.19	71.65
$M_2^{flexible+single}$	$\alpha \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$	65.0	72.44
$M_2^{flexible+combine}$	$\alpha \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$	**65.70**	**72.64**

(problem P_2). For this purpose, we tested the algorithm on the datasets of four words including *interest*, *line*, *hard*, and *serve*. All examples in these datasets were tagged with the right senses. The sizes of these datasets are 2369, 4143, 4378, and 4342, respectively. These datasets are large enough to be divided into labeled and unlabeled datasets. We randomly selected 200 examples for labeled data, 100 examples for test data, and the remaining examples are treated as unlabeled examples. Note that, because we knew the tagged senses of examples in unlabeled datasets, we are able to evaluate the correctness of the new labeled examples (for problem P_2). Fig. 3 shows experimental results of the test, in which two solutions, corresponding to using a single classifier or multiple classifiers, were investigated. As we can see in Fig. 3, using multiple classifiers in combination yields a lower classification error rate.

Table 3 shows the results for the test of P_1 and P_2 problems, in which the conventional self-training algorithm, denoted by M_0, and the models M_1 and M_2 were implemented, where NB classifier is used as the baseline. From these

Table 4
Test for Problem P_3

		Max	Min	Med	Majority vote
Sen-2	M_3^{two}	*66.27*	**66.30**	66.25	65.35
	M_3^{all}	*66.04*	**66.27**	65.74	65.70
Sen-3	M_3^{two}	*73.27*	73.23	**73.30**	72.49
	M_3^{all}	*73.22*	72.74	**73.33**	73.20

Table 5
A Comparison on Senseval-2 and Senseval-3

	Supervised Learning			M_3^{two}	
	NB	SVM	MEM	*max* rule	best rule
Senseval-2	64.05	63.72	64.79	**66.27**	**66.30**
Senseval-3	71.96	70.87	71.91	**73.27**	**73.30**

results, we have the following conclusions.

- Better results given by model M_1 in comparison with model M_0 reflect that using the procedure of retaining class distribution is effective.
- Models M_2 give better results in comparison with models M_1. This shows that the proposed solutions for problem P_2 are quite effective. In addition, using flexible determination of α integrated with a strategy of classifier combination gives the best result.
- Only model M_2 yields better results in comparison with baseline results. With the proposed solutions for P_1 and P_2, we have shown that unlabeled data can significantly improve the performance of supervised learning.

Table 4 shows the results for the test of P_3. As we have seen, the results from these two models M_3^{two} and M_3^{all} are not much different. Therefore, the model M_3^{two} should be chosen, due to the costs of computation time and storage space. Further, we also see that the max rule used for the combination of generated classifiers gives acceptable results, which obtain the best results in most cases.

In summary, Table 5 shows a comparison between the supervised learning algorithms and the selected semi-supervised learning model, namely M_3^{two}. For supervised learning, we implemented three algorithms including NB, MEM, and SVM. The obtained results show that model M_3^{two} yields better results in comparison with supervised WSD. In addition, these results are also better than the state-of-the-art systems for the English lexical sample task of Senseval-2 (see (Kilgarriff & Rosenzweig , 2000)) and Senseval-3 (see (Mihalcea, 2004b)) (66.27% in comparison with 64.2% for Senseval-2, and 73.27% in comparison with 72.9% for Senseval-3).

Table 6
p-values of t-test for the proposed bootstrapping algorithm against supervised test(using MEM)

	Algorithms	Average accuracy	t-test p-value
Senseval-2	Bootstrapping	66.22%	1.56537E-05
Senseval-3	Bootstrapping	73.25%	2.75019E-05

5.6 Significant Test and Computational Time

In order to validate the result obtained with the improved bootstrapping method, we have conducted a significant test (t-test) as shown in Table 6, where the model M_3^{two} with *max* rule for combination strategy selected as the improved bootstrapping algorithm is tested against supervised algorithm MEM. This experiment was carried out by running the selected bootstrapping model 5 times on Senseval-2 and Senseval-3, respectively. Then the obtained accuracies are used together with the accuracies of MEM algorithm taken from Table 5 to do a t-test. It has been shown from Table 6 that the improvement of the bootstrapping algorithm is confident as reported.

Concerning efficiency of the improved bootstrapping method, Table 7 shows computational time (measured by seconds) of the proposed bootstrapping algorithm and the supervised learning algorithm MEM. This experiment was done on a PC with Pentium IV 3.00GHz processor and 1GB RAM. In the table, numbers of sentences of labeled data, unlabeled data, and test data are also specified to provide a view of data's size. There are 73 and 57 ambiguous words in Senseval-2 and Senseval-3, respectively, and each ambiguous word has separately a training data and a test data. Note that computational time of the test for Senseval-2 is longer than that for Senseval-3 due to the fact that the data size of each ambiguous word in Senseval-3 is bigger than that in Senseval-2. As shown in Table 7, the bootstrapping algorithm takes much time for training. This is reasonable because of the bootstrapping algorithm aims to enlarge the labeled dataset with much more examples. In terms of the computational time in comparison with supervised learning using MEM, the bootstrapping algorithm takes about 4 times longer for Senseval-2 (106 versus 26), and about 6 times longer for Senseval-3 (191 versus 31). However, in our opinion, this computational cost may be acceptable for such a non-realtime application as preparing data for information retrieval or information extraction.

Table 7
Average computational time (measured by seconds) of the proposed bootstrapping algorithm

	Number of sentences for			Computational time	
	labeled data	unlabeled data	test data	Supervised (MEM)	Bootstrapping Training/Test
Senseval-2	8611	135705	4328	26	6895/106
Senseval-3	8522	160746	3944	31	6490/191

6 Related Work and Discussion

As a study of the bootstrapping approach for WSD considered in the present paper, it would be worth to go into more details about the previously related studies which have also applied the bootstrapping to WSD.

First, Yarowsky (1995) did use some labeled examples as seeds and extracted from them the decision rules for sense disambiguation. These decision rules were then used to detect senses for new examples based on decision list algorithm. The process of detecting new labeled examples obeys the so-called principle "one sense per collocation". As a result, new labeled examples will be added to the original set of labeled examples, and from them new decision rules are extracted. This process is repeated until the algorithm converge. In (Mihalcea, 2004a), the author carried out an investigation of applying co-training and self-training to WSD, and discussed about the three parameters of these algorithms, including: number of iterations, number of most confidently labeled examples that are added at each iteration. In particular, various setting of these parameters were validated on the training data to find the best setting. Furthermore, Mihalcea (2004a) also proposed a smoothing technique with majority voting, that is, during the bootstrapping process, the classifier at each iteration is replaced with a majority voting scheme applied to all classifiers built at previous iterations. The experiments conducted on Senseval-2 did show an improvement with the global parameter setting and the smoothing technique (average error reduction of 9.8%), and also a similar performance for both co-training and self-training. Zheng *et al.* (2005) did investigate a Label Propagation (LP) based semi-supervised learning algorithm proposed in (Zhu & Ghahramani, 2002) for WSD. The authors also suggested an entropy based method to automatically identify a distance measure that can boost the performance of LP algorithm on a given dataset. Pham *et al.* (2005) have made use of co-training in spectral graph transductive (SGT) (Joachims, 2003) for WSD in which, instead of directly computing the nearest neighbor graph in SGT, the authors constructed a separate graph for each view, and then combined them together to obtain the final graph. Their empirical test on Senseval-2 showed a better performance in comparison to co-training,

smoothed co-training (Mihalcea, 2004a), and origin SGT.

In addition, in (Abney, 2002) the author did a mathematical analysis of Yarowsky's bootstrapping algorithm considered as an optimization problem and proposed some variants for it. This approach is different from our approach in the aspect that it just focuses on the final objective function, while we directly address to particular problems which could have an impact on the quality of the objective function. However, a further study focusing on an integration of two approaches would be highly interesting to consider.

As already mentioned previously, all of these studies have not considered simultaneously all the above identified problems in a common framework of bootstrapping. In the present paper, we have taken into account these problems integratedly in the bootstrapping framework and discussed solutions for them. Regarding the novel aspect of discussed solutions, it would be worth emphasizing the following points.

- For problem P_1, we have proposed a procedure for flexibly retaining class distribution of extended labeled data, while Mihalcea (2004a) and the others fixed the number of class-based examples.
- For problem P_2, we have proposed a flexible and dynamical procedure for determining the threshold serving for the selection of new labeled examples. This has not been considered in Goldman & Zhou (2000).
- For problem P_3, while Mihalcea (2004a) did use classifier combination techniques in the step of extending labeled data, we have applied these techniques in both steps: extending labeled datasets and generating the final classifier.

Finally, all of the above considerations have been integrated simultaneously for the objective of improving semi-supervised learning.

Also, in order to have a better view of the experimental comparison between the present paper and previous work about the improvement of proposed bootstrapping classifier with respect to the basic classifier (supervised classifier trained on original labeled data), it would be helpful to mention here experimental tests conducted in previous work. Particularly, Yarowsky (1995) just tested on a polysemous word with few seed examples. Mihalcea (2004a) conducted a test on English lexical sample of Senseval-2 and improved supervised classifier from 53.84% to 55.67% (use co-training) and 54.16% (use self-training), however, only some words in the Senseval-2 dataset were tested. In (Pham *et al.*, 2005), the authors carried out a test on English lexical sample of Senseval-2 and improved the supervised classifier from 62.9% to 65.0%, and only nouns in the Senseval-2 dataset were considered. While Zheng *et al.* (2005) tested on English lexical sample of Senseval-3 and got an improvement of the supervised classifier from 69.7% to 70.3%.

As such, the improvements over the basic classifier from these previous studies are similar to our reported results. Further, because all of these studies, with an exception of Zheng *et al.* (2005), conducted their tests only on a part of Senselval-2 dataset, it is inappropriate to have a global comparison of the final results; while in the comparison to (Zheng *et al.*, 2005), our work gives a better result.

7 Conclusion

In this paper, we have identified three problems that may occur in semi-supervised learning methods, and particularly investigated them for WSD. We proposed solutions for these problems, and these solutions form the basis for developing a new bootstrapping algorithm. To test the effectiveness of the proposed solutions, we have generated various models of the new bootstrapping algorithm and tested them on Senseval-2 and Senseval-3. The experimental results show that the proposed solutions are effective for improving semi-supervised learning for WSD, and unlabeled data can significantly improve supervised WSD as well.

Acknowledgements

The authors are grateful to the anonymous reviewers for their insightful and constructive comments that have helped to significantly improve the presentation of this paper.

This research was partly conducted as a program for the "Fostering Talent in Emergent Research Fields" in Special Coordination Funds for Promoting Science and Technology by the Japanese Ministry of Education, Culture, Sports, Science and Technology. The work on this paper was also partly supported by the research project No. 204006 entitled "Modern Methods for Building Intelligent Systems" granted by the National IT Fundamental Research Program of Vietnam.

References

S. Abney, 2002. Bootstrapping. In *Proc. ACL*, pages 360–367.

D. Angluin, and P. Laird, 1988. Learning from noisy examples. *Machine Learning*, Vol. 2, pages 343–370.

A. Blum, and T. Mitchell, 1998. Combining labeled and unlabeled data with co-training. *Proceedings of COLT*, pages 92–100.

A. Blum and S. Chawla, 2001. Learning from Labeled and Unlabeled Data Using Graph Mincuts. *Proceedings of ICML*, pages 19–26.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, 1991. Word Sense Disambiguation Using Statistical Methods. *Proceedings of ACL*, pages 264–270.

M. Collins, Y. Singer, 1999. Unsupervised Models for Named Entity Classification. *Proceedings of EMNLP*, pages 100–111.

I. Dagan and A. Itai. 1994. Word Sense Disambiguation Using a Second Language Monolingual Corpus. *Computational Linguistics*, Vol. 20(4), pages 563–596.

M. Diab and P. Resnik, 2002. An Unsupervised Method for Word Sense Tagging Using Parallel Corpora. *Proceedings of ACL*, pages 255–262.

N. Ide and J. Véronis, 1998. Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics* Vol. 24, pages 1–40.

T. Joachims, 1999. Transductive Inference for Text Classification Using Support Vector Machines. *Proceedings of ICML*, pages 200–209.

T. Joachims, 2003. Transductive Learning via Spectral Graph Partitioning. *Proceedings of ICML*, pages 290–297.

Y. Karov and S. Edelman, 1998. Similarity-Based Word Sense Disambiguation. *Computational Linguistics*, Vol. 24(1), pages 41–59.

A. Kilgarriff, J. Rosenzweig, Framework and results for English SENSEVAL, *Computers and the Humanities* **36** (2000) 15–48.

J. Kitter, M. Hatef, R. Duin and J. Matas, 1998. On combining Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2(3), pages 226–239.

A. Kilgarriff, 2002. English Lexical Sample Task Description, *Proceedings of Senseval-2 Workshop, ACL*, pages 17–20.

L. I. Kuncheva, 1997. An application of OWA operators to the aggregation of multiple classification decisions. In R. R. Yager and J. Kacprzyk (Eds.), *The Ordered Weighted Averaging Operators. Theory and Applications*, Kluwer Academic Publishers, pp. 330–343.

L. I. Kuncheva, 2001. Combining classifiers: Soft computing solutions, in: S. K. Pal, A. Pal, *Pattern Recognition: From Classical to Modern Approaches*, World Scientific, 2001, pp. 427–451.

S. Goldman and Y. Zhou, 2000. Enhancing supervised learning with unlabeled data. *Proceedings of ICML*, pages 327–334.

C. A. Le, V.N. Huynh, H. C. Dam, A. Shimazu, 2005. Combining Classifiers Based on OWA Operators with an Application to Word Sense Disambiguation. In D. Ślęzak *et al.* (Eds.): *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing – RSFDGrC 2005*, Springer-Verlag, Berlin Heidelberg 2005, pp. 512–521.

C. Leacock, G. Miller, and M. Chodorow, 1998. Using Corpus Statistics

and WordNet Relations for Sense Identification. *Computational Linguistics*, 24(1), pages 147–165.

Y. K. Lee and H. T. Ng, 2002. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. *Proceedings of EMNLP*, pages 41–48.

H. Li and C. Li, 2004. Word Translation Disambiguation Using Bilingual Bootstrapping. *Computational Linguistics*, Vol. 30(1), pages 1–22.

D. K. Lin, 1997. Using Syntactic Dependency as Local Context to Resolve Word Sense Ambiguity. *Proceedings of ACL*, pages 64–71.

M. Lesk, 1986. Automated Word Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. *Proceedings of the ACM SIGDOC Conference*.

D. McCarthy, R. Koeling, J. Weeds, and J. Carroll, 2004. Finding Predominant Word Senses in Untagged Text. *Proceedings of ACL*, pages 279–286.

R. Mihalcea, 2004a. Co-training and Self-training for Word Sense Disambiguation. *Proceedings of CoNLL*, pages 33–40.

R. Mihalcea , T. Chklovski, and A. Killgariff, 2004b. The Senseval-3 English Lexical Sample Task, *Proceedings of ACL/SIGLEX Senseval-3*, pages 25–28.

D. J. Miller and H. S. Uyar, 1997. A Mixture of Experts Classifier with Learning Based on Both Labelled and Unlabelled Data. *Advances in Neural Information Processing Systems* 9, M. Mozer, M. I. Jordan, and T. Petsche (Eds.), Cambridge, MA: MIT Press, pages 571–577.

K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, 2000. Text Classification from Labeled and Unlabeled Documents Using EM. *Machine Learning*, vol. 39 (2), pages 103–134.

H. T. Ng, B. Wang, and Y. S. Chan, 2003. Exploiting Parallel Texts for Word Sense Disambiguation: An Empirical Study. *Proceedings of ACL*, pages 455–462.

H. T. Ng and H. B. Lee, 1996. Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach. *Proceedings of ACL*, pages 40–47.

M. P. Perrone and L. N. Cooper, 1993. When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. *Neural Networks for Speech and Image Processing*, pages 126–142.

T. P. Pham, H. T. Ng, W. S. Lee, 2005. Word Sense Disambiguation with Semi-Supervised Learning. *Proceedings of AAAI*, pages 1093–1098.

D. Pierce and C. Cardie, 2001. Limitations of Co-Training for Natural Language Learning from Large Datasets. *Proceedings of EMNLP*, pages 1–9.

R. R. Yager, 1988. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision Making, *IEEE Transactions on Systems, Man, and Cybernetics* Vol. 18, pages 183–190.

W. Su, M. Carpuat, and D. Wu, 2004. Semi-Supervised Training of a Kernel PCA-Based Model for Word Sense Disambiguation. *Proceedings of COLING*, pages 1298–1304.

H. C. Seo, H. J. Chung, H. C. Rim, S. H. Myaeng, and S. H. Kim, 2004. Unsu-

pervised Word Sense Disambiguation Using WordNet Relatives. *Computer, Speech and Language*, Vol. 18(3), pages 253–273.

Z.-H. Zhou , M. Li, 2005. Tri-Training: Exploiting Unlabeled Data Using Three Classifiers. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17(11), pages 1529–1541.

Y. Tsuruoka and J. Tsujii, Chunk Parsing Revisited. In Proceedings of the 9th International Workshop on Parsing Technologies (IWPT 2005), pp. 133–140

D. Yarowsky, 1992. Word Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. *Proceedings of COLING*, pages 454–460.

D. Yarowsky, 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. *Proceedings of ACL*, pages 189–196.

Y. N. Zheng, H. J. Dong, and L. T. Chew, 2005. Word Sense Disambiguation Using Label Propagation Based Semi-supervised Learning Method. *Proceedings of ACL*, pages. 395–402.

X. Zhu and Z. Ghahramani, 2002. Learning from Labeled and Unlabeled Data with Label Propagation. *CMU CALD tech report* CMU–CALD–02–107.