Supervised and Unsupervised Learning Algorithms for Thai Web Pages Identification

Boonserm Kijsirikul[1], Puay Sasiphongpairoege[1], Nuanwan Soonthornphisaj[1], and Surapant Meknavin[2]

[1] Department of Computer Engineering, Chulalongkorn University, Phathumwan, Bangkok, 10330, Thailand.
email: {boonserm,puay,nuanwan}@mind.cp.eng.chula.ac.th
[2] Siamguru Co.,Ltd. 2922/103 Charn Issara Tower II, 126-7 New Petchburi Rd., Bangkapi, Huay Kwang, Bangkok 10310, Thailand.
email: surapan@siamguru.com

Abstract. The paper presents a learning method, called *iterative cross-training (ICT)* for identifying Thai Web pages. Our method combines two classifiers, i.e. a word segmentation classifier and a naive Bayes classifier, that use unlabeled examples to train each other. We compare ICT against other supervised and unsupervised learning methods: a supervised word segmentation classifier *(S-Word)*, a supervised naive Bayes classifier *(S-Bayes)*, an unsupervised naive Bayes classifier using the EM algorithm *(U-Bayes-EM)*, and a co-training-style classifier *(CoTraining)*. The experimental results show that ICT gives the best performance, followed by S-Bayes, CoTraining U-Bayes-EM and S-Word.

1 Introduction

Given pre-labeled training data, supervised learning has been successfully applied to text classification [1, 5, 6, 3, 10] However, one of the difficulties of using supervised learning is that we have to hand-label data for constructing training sets. Though it is costly to construct hand-labeled data, in some domains it is easy to obtain unlabeled ones, such as data in the World Wide Web. Thus, if we are able to effectively utilize the available unlabeled data, we will simplify the task of building text classifiers. Various methods have been proposed to use unlabeled data together with pre-labeled data for text classification, such as text classification using the EM algorithm [9], the co-training algorithm [2].

This paper describes our work that is a part of our project on building a system which retrieves information from the Web. The goal of our project is to build a Web robot that crawls the Web and determines if a page is of interest. In this paper, we focus on a method for classifying Web pages into the set of Thai pages and the set of non-Thai pages. In fact, we want our Web robot to retrieve only Thai pages. By a Thai page, we refer to the page that is intended to be read by Thai people and contains Thai texts and may contain texts in other languages.

We propose a method, called *iterative cross-training (ICT)*, for identifying Thai Web pages. Our method is an unsupervised learning in the sense that it needs no pre-labeled examples and thus it is suitable for this domain where unlabeled data is plentiful and easy to obtain. The method combines two classifiers which iteratively train each other for improving the performance of the classifiers. Given two sets of unlabeled data, each of which is for each classifier, the classifiers label the data for the other. The first classifier is given some knowledge about the domain, and uses the knowledge to estimate labels of the pages for the second classifier. The first classifier may require expensive computation. The second classifier has no domain knowledge and requires inexpensive computation. It learns its model from pages labeled by the first, and uses the current model to label training data for the first. This cross-training process is iterated. The expensive classifier used in our task is a word segmentation classifier that is given knowledge in a form of dictionary. The inexpensive one is a naive Bayes classifier. With good interaction between two classifiers, the performance of the whole system is increasingly improved. After the classifiers are trained, only one classifier will be used by the Web robot. It is desirable to use the inexpensive one for fast retrieval of Thai Web pages. In case that the accuracy, not classification time, is the main concern, the expensive one can be used as well. One of the advantages of our method is that, as the method does not require human to label the Web pages, it can be trained with a lot of unlabeled pages and can be easily changed for identifying other languages.

To evaluate the effectiveness of our method, we implement other four classifiers to compare empirically with our method. The implementation is designed to explain, or at least give some answers to questions: "is iterative cross-training (ICT) which combines two classifiers an effective method?", "does this kind of combination of two classifiers perform better than only one?", and "can the method successfully use unlabeled data?". The other four classifiers are: (1) a supervised word segmentation classifier (S-Word), (2) a supervised naive Bayes classifier (S-Bayes), (3) a co-training-style classifier (CoTraining), and (4) an unsupervised naive Bayes classifier using the EM algorithm (U-Bayes-EM).

The experimental results show that ICT successfully and efficiently identifies Thai Web pages. The overall performance, evaluated by F_1-measure, of ICT is the best. The better performance of ICT over U-Bayes-EM, which uses single classifier, shows the effectiveness of the combination of two classifiers. The comparable performance of our method to supervised ones (S-Bayes and S-Word) demonstrates the successful use of unlabeled data.

2 Iterative Cross-Training

This section presents the iterative cross-training. First we describe the architecture of our learning system, and then give the details of two classifiers used in the system. Figure 1 shows our learning system which learns to classify Web pages. The system is composed of two classifiers: (1) a word segmentation classifier and (2) a naive Bayes classifier. These two classifiers estimate their parameters from unlabeled data by receiving training from each other. Two training data sets, called $TrainingData1$ and $TrainingData2$ are selected from the training data provided by user. These two training sets may overlap, be identical or different. Let θ_w and θ_n be sets of parameters of the word segmentation classifier and of the

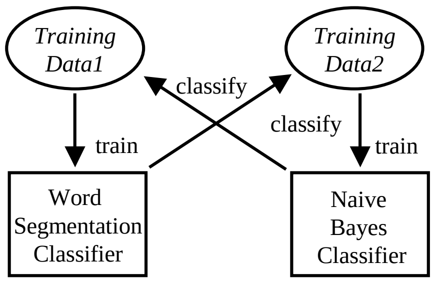

Fig. 1. The architecture of iterative cross-training. It is composed of two classifiers which use unlabeled data to train each other.

Table 1. The training algorithm of iterative cross-training.

Given:

- two sets $TrainingData1$ and $TrainingData2$ of unlabeled training examples

Initialize the parameter set of the word segmentation classifier to θ_{w0} ($\theta_w \leftarrow \theta_{w0}$).
Loop until θ_w does not change or the number of iterations exceeds a predefined value:

- Use the word segmentation classifier with the current parameter set θ_w to classify $TrainingData2$ into positive examples and negative examples.
- Train the naive Bayes classifier by the previously labeled $TrainingData2$ to estimate the parameter set θ_n of the classifier.
- Use the naive Bayes classifier with the current θ_n to classify $TrainingData1$ into positive examples and negative examples.
- Use the previously labeled $TraingingData1$ to determine the parameter set θ_w of the word segmentation classifier.

naive Bayes classifier, respectively. $TrainingData1$ is used for training the word segmentation classifier to estimate its parameter set, and the $TrainingData2$ is used for estimating the parameter set of the naive Bayes classifier. The algorithm for training the classifiers is shown in Table 1.

The idea behind our algorithm is that if we can obtain reliable statistical information contained in $TrainingData2$, it should be useful in classifying $TrainingData1$. If the starting θ_{w0} has property that it produces more true positive than wrong positive examples and more true negative than wrong negative examples for $TrainingData2$, the statistical information in correctly classified examples will be obtained. Using this information the naive Bayes classifier should classify more correct examples in $TrainingData1$ that have similar characteristics. If the newly labeled $TrainingData1$ can produce θ_w better than θ_{w0}, more reliable parameters of the whole system should be obtained after each iteration. We will discuss the property of θ_{w0} that produces more correct examples in Section 2.1. The following subsections describe the details of the classifiers.

2.1 Word Segmentation Classifier

One straightforward way to determine whether a Web page is in a specific language is to check the words in the page with a dictionary. If many words appear in the dictionary, it is likely that the page is in that language. We cannot expect that all words in the page appear in dictionary as the Web page usually contains names of persons, organizations, etc. not occurring in the dictionary and may contain words written in foreign languages. Therefore, it is necessary to determine how many words should be contained. This task is more difficult when it is considered in a language that has no word boundary delimiters, such as Thai, Japanese, etc. [7]. Below we describe our method for word segmentation.

Given a dictionary and a document d of n characters $(c_1, c_2, \ldots, c_n)$, the word segmentation classifier generates all possible segmentations and finds the best segmentation $(w_1, w_2, \ldots, w_m)$ that minimizes the cost function in Equation 1.

$$\text{argmin}_{w_1,\ldots,w_m} \sum_{i=1}^{m} cost(w_i) \tag{1}$$

$$\text{where } cost(w_i) = \begin{cases} \eta_1 \text{ if } w_i \text{ is a word in the dictionary} \\ \eta_2 \text{ if } w_i \text{ is a string not found in the dictionary} \end{cases}$$

In our experiments, η_1 and η_2 are set to 1 and 2, respectively. Any sequence of characters, $c_i, \ldots, c_j$, found in the dictionary must be considered as a word, and must not be grouped with nearby characters to form a long unknown string.

After the best segmentation is determined, the document is composed of (1) words appeared in the dictionary, and (2) unknown strings not found in the dictionary. A Thai Web page should be the page that contains many words and few unknown strings. We then define $WordRatio$ of a page as:

$$\frac{\text{the number of characters in all words}}{\text{the number of all characters in the document}}$$

Given sets of positive and negative examples, the classifier finds the threshold of $WordRatio$ that maximizes the number of correctly classified positive and negative examples. If $WordRatio$ of a page is greater than the threshold, we will classify the page as positive (Thai page). Otherwise, we will classify it as negative (non-Thai page). For simplicity, let us use only the threshold of $WordRatio$ as the parameter of the word segmentation classifier (θ_w). Having only the threshold of $WordRatio$ (θ_w) as the parameter, we can find θ_{w0} which produces more true positive and true negative examples for $TrainingData2$. As described above, most of Thai pages should have a high value of $WordRatio$, whereas non-Thai pages should have a low value one. If the numbers of Thai and non-Thai pages in $TrainingData2$ are the same, it is easily seen that any value of θ_{w0} will give more correctly classified pages than incorrect ones (except for θ_{w0}=0.0 or θ_{w0}=1.0, that gives the same number of correctly and incorrectly classified pages). In case that the number of Thai pages is lower than the number of non-Thai pages, a high value of θ_{w0}, (e.g. 0.7, 0.8, 0.9) will produce more correctly classified pages. A low value of θ_{w0} is for the case that the number of Thai pages is larger than non-Thai pages.

A new θ_w can be estimated, after the naive Bayes classifier labels data in $TrainingData1$. Let SP be the smallest value of $WordRatio$'s from all labeled positive examples, and LN be the largest value from all labeled negative examples. The new θ_w is estimated as:

$$\theta_w = (SP + LN)/2 \tag{2}$$

2.2 The Naive Bayes Classifier

For text classification, naive Bayes is among the most commonly used and the most effective methods [8]. To represent text documents, the method usually employs *bag-of-words* representation. Instead of bag-of-words, we use the simpler *bag-of-characters* representation. This representation is suitable for a Web robot to identify Thai Web pages, because it requires no word segmentation and thus it is very fast. In spite of its simplicity, our results show the effectiveness of bag-of-characters representation in classifying Web pages, as shown later in Section 4.

Given a set of class labels $L = \{l_1, l_2, \ldots, l_m\}$ and a document d of n characters $(c_1, c_2, \ldots, c_n)$, the most likely class label l^* estimated by naive Bayes is the one that maximizes $Pr(l_j|c_1, \ldots, c_n)$:

$$\begin{aligned} l^* &= \text{argmax}_{l_j} Pr(l_j|c_1, \ldots, c_n) && (3) \\ &= \text{argmax}_{l_j} \frac{Pr(l_j)Pr(c_1, \ldots, c_n|l_j)}{Pr(c_1, \ldots, c_n)} && (4) \\ &= \text{argmax}_{l_j} Pr(l_j)Pr(c_1, \ldots, c_n|l_j) && (5) \end{aligned}$$

In our case, L is the set of positive and negative class labels. $Pr(c_1, \ldots, c_n)$ in Equation 4 can be ignored, as we are interested in finding the most likely class label. As there are usually an extremely large number of possible values for $d = (c_1, c_2, \ldots, c_n)$, calculating the term $Pr(c_1, c_2, \ldots, c_n|l_j)$ requires a huge number of examples to obtain reliable estimation. Therefore, to reduce the number of required examples and improve reliability of the estimation, assumptions of naive Bayes are made [8]. These assumptions are (1) the conditional independent assumption, i.e. the presence of each character is conditionally independent of all other characters in the document given the class label, and (2) an assumption that the position of a character is unimportant, e.g. encountering the character "a" at the beginning of a document is the same as encountering it at the end. Using the above assumptions, Equation 5 can be rewritten as:

$$\begin{aligned} l^* &= \text{argmax}_{l_j} Pr(l_j) \prod_{i=1}^{n} Pr(c_i|l_j, c_1, \ldots, c_{i-1}) && (6) \\ &= \text{argmax}_{l_j} Pr(l_j) \prod_{i=1}^{n} Pr(c_i|l_j) && (7) \end{aligned}$$

This model is also called unigram model because it is based on statistics about single character in isolation. The probabilities $Pr(l_j)$ and $Pr(c_i|l_j)$ are used as the parameter set θ_n of our naive Bayes classifier, and are estimated from the

Table 2. The co-training-style algorithm.

```
Given:
  • a set $LE$ of labeled training examples
  • a set $UE$ of unlabeled examples
Create a pool $UE'$ of examples by choosing $u$ examples at random from $UE$.
Loop until no examples left in $UE$:
  – Use $LE$ to estimate $\theta_w$ of the word segmentation classifier.
  – Use $LE$ to estimate $\theta_n$ of the naive Bayes classifier.
  – Allow the word segmentation classifier with $\theta_w$ to label $p$ positive and $n$ negative
    examples from $UE'$.
  – Allow the naive Bayes classifier with $\theta_n$ to label $p$ positive and $n$ negative examples
    from $UE'$.
  – Add these self-labeled examples to $LE$.
  – Randomly choose $2p + 2n$ examples from $UE$ to replenish $UE'$.
```

training data. The prior probability $Pr(l_j)$ is estimated as the ratio between the number of examples belonging to the class l_j and the number of all examples. The conditional probability $Pr(c_i|l_j)$, of seeing character c_i given class label l_j, is estimated by the following equation:

$$Pr(c_i|l_j) = \frac{1 + N(c_i, l_j)}{T + N(l_j)} \tag{8}$$

Where $N(c_i, l_j)$ is the number of times character c_i appears in the training examples from class label l_j, $N(l_j)$ is the total number of characters in the training set for class label l_j, and T is the total number of unique characters in the training set. Equation 8 employs Laplace smoothing (adding one to all the character counts), to avoid assigning probability values of zero to characters that do not occur in the training examples for a particular class.

3 Other Classifiers Used in Comparison

3.1 Co-Training-Style Classifier

The co-training algorithm is described in [2]. The idea of the algorithm is that an example can be considered in two different views, and either view is assumed to be sufficient for learning. Based on this idea, we construct a co-training-style algorithm for our task. The algorithm is shown in Table 2. To apply this idea to our problem, we view each Web page as (1) a set of characters occurring in the page, and (2) a set of words occurring in that page. A naive Bayes classifier is employed to learn from the view of the character representation, and a word segmentation classifier is used for the word representation. The parameters θ_w and θ_n are estimated in the same way as described in Section 2. The algorithm requires a small set of hand-labeled data for beginning the training process. Therefore, we can think of this algorithm as semi-supervised one.

Table 3. Training algorithm for the naive Bayes classifier using the EM algorithm.

Given:
- a set UE of unlabeled examples

Use the word segmentation classifier with initial θ_{w0} to label UE.
Use the labeled examples in UE to estimate the parameters $Pr(c_i|l_j)$ and $Pr(l_j)$ of the naive Bayes classifier with $Pr(l_j|d) \in \{0,1\}$.
Loop until the parameters of Bayes do not change or the number of iterations exceeds a predefined value:

– (E-step) Estimate the probabilistically-weighted class labels, $Pr(l_j|d)$, for every document using Equation 11.
– (M-step) Use the estimated class labels, $Pr(l_j|d)$, to calculate new paramenters using all documents, by Equation 9 and 10.

3.2 Unsupervised Naive Bayes Classifier Using the EM Algorithm

This subsection describes an unsupervised naive Bayes classifier which uses Expectation-Maximization(EM) algorithm [4] for filling the missing class labels in training examples. Our method for training naive Bayes with the EM algorithm is the same as one described in [9], and is shown below.

Let $L = \{l_1, l_2, \ldots, l_m\}$ be a set of class labels, d be a document of n characters $(c_1, c_2, \ldots, c_n)$ from a data set D, $Pr(l_j|d) \in \{0,1\}$ be the class label of the document d. The estimate of the probability of character c_i in class label l_j is:

$$Pr(c_i|l_j) = \frac{1 + \sum_{d \in D} N(c_i, d) Pr(l_j|d)}{T + \sum_{k=1}^{T} \sum_{d \in D} N(c_k, d) Pr(l_j|d)} \tag{9}$$

Where T is the total number of unique characters in the training set, $N(c_i, d)$ is the number of times character c_i occurs in document d. The probability of a class label is given by Equation 10:

$$Pr(l_j) = \frac{1 + \sum_{d \in D} Pr(l_j|d)}{|L| + |D|} \tag{10}$$

Where $|L|$, $|D|$ are the number of class labels, and the number of documents in the training set. Given an unlabeled document d, of n character $(c_1, c_2, \ldots, c_n)$, the naive Bayes classifier estimates the probability that the document belongs to the class label l_j by using Equation 11 below.

$$Pr(l_j|d) = \frac{Pr(l_j)Pr(d|l_j)}{Pr(d)} = \frac{Pr(l_j) \prod_{i=1}^{n} Pr(c_i|l_j)}{\sum_{k=1}^{|L|} Pr(l_k) \prod_{i=1}^{n} Pr(c_i|l_k)} \tag{11}$$

Note that now $Pr(l_j|d)$ is a probabilistically-weighted value; each document d is considered to be of class label l_j with probability equal to the estimated $Pr(l_j|d)$. The algorithm for training the naive Bayes classifier using the EM algorithm is shown in Table 3. The algorithm uses the word segmentation classifier once for determining the initial labels for the training data.

4 Experimental Results

We conducted experiments to compare iterative cross-training (ICT) with the other four classifiers described in the previous section: the supervised word segmentation classifier (S-Word), the supervised naive Bayes classifier (S-Bayes), the co-training-style classifier (CoTraining), and the unsupervised naive Bayes classifier using the EM algorithm (U-Bayes-EM). S-Word and S-Bayes used in our comparison are the same as ones described in Section 2.1 and 2.2, except that they are trained by hand-labeled data.

4.1 Data Set & Experimental Setting

We collected the data set for our experiments by starting from four Web pages: a Japanese Web page, two Thai Web pages, and an English web page. From each of these four pages, a Web robot is used to recursively follow the links within the page until it retrieves 450 pages. Therefore, we have approximately 900 Thai pages as Thai pages may link to ones which are in English or other languages. We also have approximately 450 Japanese and 450 English pages. All of these pages were divided into three sets, denoted as A, B and C, each of which contains 600 pages (about 300 Thai, 150 Japanese and 150 English pages). We used 3-fold cross validation in all experiments below for averaging the results. The following are the parameter settings for the classifiers when C was used as the test set.

(1) For ICT, $A+B$ was used as both $TrainingData1$ and $TrainingData2$. The initial θ_{w0} was set to 0.7.

(2) For each of S-Word and S-Bayes classifiers, the pages in sets A and B were manually labeled and the experiment was run for two times; the first run with the labeled A and the second run with the labeled B as the training set. The results on classifying C were averaged.

(3) For CoTraining, $A+B$ was used as the training set. The values of the parameters of the classifier (in Table 2) were set in a similar way as in [2]. As CoTraining requires a small set of correctly pre-classified training data, we gave the algorithm with 18 hand-labeled pages. In our experiment, we set the values of $|LE|$, $|UE|$, p, n and u to 18, 1182, 3, 3 and 115, respectively.

(4) For U-Bayes-EM, $A+B$ was used as the training set. The initial θ_{w0} for the algorithm (in Table 3) was set to 0.7.

4.2 The Results

To evaluate the performance of the classifiers, we use standard precision(P), recall(R) and F_1-measure(F_1) defined as follows:

$$P = \frac{\text{no. of correctly predicted positive examples}}{\text{no. of predicted positive examples}}$$

$$R = \frac{\text{no. of correctly predicted positive examples}}{\text{no. of all positive examples}}$$

Table 4. The precision(P), recall(R) and F_1-measure(F_1) of the classifiers.

Classifier	P	R	F_1
ICT(Word)	99.78	100.00	99.89
S-Bayes	100.00	99.00	99.50
ICT(Bayes)	100.00	98.89	99.44
CoTraining(Bayes)	100.00	98.89	99.44
U-Bayes-EM	100.00	98.78	99.39
S-Word	99.08	99.61	99.34
CoTraining(Word)	100.00	98.66	99.33

$$F_1 = \frac{2PR}{P+R}$$

The results are shown in Table 4. In the table, "CoTraining(Bayes)" and "CoTraining(Word)" are the results of the naive Bayes and the word segmentation classifiers of CoTraining, respectively. "ICT(Bayes)" and "ICT(Word)" are for the naive Bayes and the word segmentation classifiers of ICT.

As shown in the table, ICT(Word) gives the best performance. S-Bayes is the second best classifier, followed by ICT(Bayes) and CoTraining(Bayes) according to F_1-measure. Among the classifiers tested in the experiments, ICT and U-Bayes-EM are unsupervised classifiers. The reason for better performance of ICT over U-Bayes-EM may be because ICT employs two sub-classifiers which help each other in learning while U-Bayes-EM uses only single classifier. Compared to supervised classifiers, the performance of ICT is comparable to that of S-Bayes and better than that of S-Word. The results demonstrate that our system can effectively use unlabeled examples and the two classifiers succeed in training each other. The training technique of ICT is also an effective one as its performance is better than that of CoTraining which uses a different training technique.

4.3 The Effect of Parameter Settings on the Performances of the Classifiers

This subsection shows additional experiments that were conducted to see the effect of parameter settings on ICT, U-Bayes-EM and CoTraining. For ICT and U-Bayes-EM, θ_{w0} is only the initial parameter that may effect the performance of the classifiers. We run experiments with different θ_{w0} varying from 0.0, 0.1, 0.2,. . ., 1.0. The results, when C was used as the test set, are shown in Table 5. When A or B was used as the test set, the similar results were obtained.

The results in the table show that each of θ_{w0} varying from 0.0 to 1.0 did not effect the precision and recall of ICT; with all of them the precision and recall were 100% and 98.33%, respectively. Each of these initial settings converged to the same final θ_w (0.301) The θ_{w0}'s which quickly converged were 0.2, 0.3, 0.4 and 0.5; each of them was close to the final θ_w. U-Bayes-EM is more sensitive to initial θ_{w0}. In case of θ_{w0} equal to 0.9 or 1.0, the algorithm converged to a local maximum which gave the precision and recall of 0%. When θ_{w0} was set to 0.0 or 0.1, U-Bayes-EM increased recall to 99.33% but decreased precision to

Table 5. The precision(P), recall(R), and number of iterations(I) before the convergence of ICT (Bayes) and U-Bayes-EM using different θ_{w0}.

θ_{w0}	ICT(Bayes)			U-Bayes-EM		
	P(%)	R(%)	I	P(%)	R(%)	I
0.0	100.00	98.33	5	66.08	99.33	3
0.1	100.00	98.33	3	66.08	99.33	4
0.2	100.00	98.33	2	100.00	98.33	3
0.3	100.00	98.33	2	100.00	98.33	3
0.4	100.00	98.33	2	100.00	98.33	4
0.5	100.00	98.33	2	100.00	98.00	5
0.6	100.00	98.33	3	100.00	98.00	6
0.7	100.00	98.33	4	100.00	98.00	7
0.8	100.00	98.33	5	100.00	98.00	11
0.9	100.00	98.33	3	0.00	0.00	6
1.0	100.00	98.33	3	0.00	0.00	9

66.08%. The reason for the better performance of ICT(Bayes) is because of the good interaction between the naive Bayes and the word segmentation classifiers of ICT.

We also run experiments to see the effect of parameter settings for CoTraining. As shown in Table 2, there are several parameters to be set, i.e., $|LE|$, p, n and u. To restrict the number of experiments, we varied only p and n, and left the other unchanged. The parameters p and n control the amount of self-labeled examples which will be added into the labeled training set. If the values of the parameters are large, the algorithm will rapidly add many self-labeled examples into the training set. We expected that the performance of CoTraining would decrease with increasing p and n. However, the results were not as expected as shown in Table 6.

Table 6. The precision(P), recall(R) and F_1-measure(F_1) of CoTraining as we vary p and n. All experiments used 3-fold cross validation.

Parameter	CoTraining(Word)			CoTraining(Bayes)		
setting	P(%)	R(%)	F_1	P(%)	R(%)	F_1
$p = n = 3$	100.00	98.66	99.33	100.00	98.89	99.44
$p = n = 6$	100.00	99.00	99.50	100.00	98.88	99.44
$p = n = 12$	100.00	98.77	99.38	100.00	98.88	99.44
$p = n = 24$	93.73	99.22	96.40	95.92	98.88	97.38
$p = n = 48$	100.00	98.55	99.27	100.00	98.88	99.44

The results show that CoTraining, especially CoTraining(Word), is sensitive to the parameter settings. Comparing the results in Table 5 and Table 6, we can see that ICT is more robust to the parameter settings and this is an advantage of ICT over U-Bayes-EM and CoTraining.

5 Conclusion

We have presented a method that effectively uses unlabeled examples to estimate the parameters of the system for classifying Web pages. The method is based on two components, i.e. the word segmentation classifier and the naive Bayes classifier, that train each other. Without the help of human in labeling the examples, the naive Bayes classifier of our system gives 100% precision and 98.89% recall of Thai Web pages tested in our experiments. This performance is competitive with those of supervised ones (S-Bayes and S-Word), which demonstrates the successful use of unlabeled data of our method. The performance is also better than that of U-Bayes-EM, which uses single classifier. The better performance of our method than U-Bayes-EM shows the effectiveness of interaction between two classifiers. Another advantage of our method over CoTraining and U-Bayes-EM is the ease of the initial parameter setting because our method is robust to the setting whereas CoTraining and U-Bayes-EM are more sensitive to the setting.

Despite simple model which uses only the threshold (θ_w) of $WordRatio$ for the word segmentation classifier and the character-unigram for the naive Bayes classifer, the high precision and recall are obtained. However, some Thai pages cannot be detected by this model. This is a limitation of the naive Bayes classifier of our current system. To improve the system, we are exploring more sophisticated models, e.g. a character or word n-gram model.

References

1. Apte, C., and Damerau, F. (1994) Automated learning of decision rules for text categorization. ACM TOIS 12(2):233-251.
2. Blum, A. and Mitchell, T. (1998) Combining labeled and unlabeled data with co-training. Proceeding of the Eleventh Annual Conference on Computational Learning Theory.
3. Cohen, W. W. and Singer, Y. (1999) Context-sensitive learning methods for text categorization. ACM Transactions on Information Systems, Vol. 17, No. 2, 141-173.
4. Dempster, A. P., Laird, N. M., and Rubin D. B. (1977) Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B, 39 (1), 1-38.
5. Joachims, T. (1998) Text categorization with support vector machines: Learning with many relevant features. Proceedings Tenth European Conference on Machine Learning, Springer Verlag.
6. Lewis, D. (1998) Naive (Bayes) at forty: The independence assumption in information retrieval. Proceedings of the Tenth European Conference on Machine Learning.
7. Meknavin, S., Charoenpornsawat, P. and Kijsirikul, B. (1997) Feature-based Thai word segmentation. Proceeding of Natural Language Processing Pacific Rim Symposium '97.
8. Mitchell, T. (1997) Machine Learning. pp. 180-184, McGraw-Hill. New York.
9. Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (2000) Text classification from labeled and unlabeled documents using EM. Machine Learning 39(2/3):103-134.
10. Yang, Y. (1999) An evaluation of statistical approaches to text categorization, Information Retrieval Journal.