งานวิจัยเรื่อง การทำเหมืองเว็บไทยโดยเทคนิคการเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัยนี้ มุ่งเน้นที่จะสร้างระบบค้นหาที่สามารถจัดหมวดหมู่ของเว็บเพจภาษาไทยให้ได้อย่างอัตโนมัติ ซึ่งเทคนิคที่จะนำมาใช้คือ การเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัย ผู้วิจัยได้ทำการวิจัยโดยเริ่มจากการดำเนินการวิจัยพื้นฐานเกี่ยวกับการโปรแกรมตรรกะเชิงอุปนัย ซึ่งจะนำไปประยุกต์ใช้กับการจำแนกเว็บเพจให้เป็นหมวดหมู่ต่อไป

ในการจำแนกเว็บเพจออกเป็นหมวดหมู่โดยอัตโนมัตินั้น สามารถทำได้หลายวิธีการด้วยกัน วิธีการหนึ่งที่น่าสนใจ คือ การโปรแกรมตรรกะเชิงอุปนัยหรือไอแอลพี (Inductive Logic Programming – ILP) เนื่องจากวิธีการนี้ ผู้สอนสามารถป้อนความรู้เบื้องต้นในรูปแบบของโปรแกรมตรรกะอันดับที่หนึ่งได้ ซึ่งจะช่วยให้การจำแนกข้อมูลทำได้อย่างมีประสิทธิภาพยิ่งขึ้น โดยทั่วไปไอแอลพีมักถูกนำไปใช้กับปัญหาที่มีลักษณะเป็นสองกลุ่ม (two-class concept) มีจุดหมายเพื่อจำแนกตัวอย่างออกเป็นสองกลุ่ม (class) คือ กลุ่มตัวอย่างบวกและกลุ่มตัวอย่างลบ โดยการสร้างกฎเพื่ออธิบายกลุ่มตัวอย่างบวก ดังนั้น เมื่อนำกฎที่สร้างได้ไปจำแนกตัวอย่างใหม่ ตัวอย่างที่ตรงกับกฎจะถูกจำแนกเป็นกลุ่มตัวอย่างบวก ส่วนตัวอย่างที่ไม่ตรงกับกฎจะถูกจำแนกเป็นกลุ่มตัวอย่างลบ แต่ในกรณีที่ต้องการนำระบบไอแอลพีไปใช้จำแนกตัวอย่างออกเป็นหลายกลุ่ม (multi-class concept) นั้น อาจมีตัวอย่างบางตัวโดยเฉพาะตัวอย่างที่มีสัญญาณรบกวน (noisy data) ที่ไม่ตรงกับกฎข้อใดข้อหนึ่งในเซตของกฎทั้งหมด ซึ่งในกรณีเช่นนี้ระบบไอแอลพีแต่เพียงอย่างเดียวไม่สามารถจำแนกตัวอย่างในลักษณะนี้ได้ จึงจำเป็นต้องมีวิธีการอื่นเข้ามาช่วยเพื่อทำให้ระบบไอแอลพีสามารถจำแนกตัวอย่างออกเป็นหลายกลุ่มได้

ดังนั้นในงานวิจัยนี้จึงเป็นการมุ่งศึกษาหาวิธีการ ซึ่งสามารถนำมาใช้จำแนกตัวอย่างเพื่อใช้ร่วมกับกฎที่ได้จากระบบไอแอลพี เราได้ใช้กระบวนการดึงลักษณะสำคัญและวิธีการ

5.4

แบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก (Backpropagation Neural Network: BNN) เพื่อประมาณกฎสำหรับเลือกกลุ่มที่ใกล้เคียงในกรณีดังกล่าว โดยใช้กระบวนการดึงลักษณะสำคัญ เพื่อหาลักษณะสำคัญจากกฎลำดับที่หนึ่ง รูปแบบของปัญหาเดิมจะเปลี่ยนไปอยู่ในรูปของค่าคุณลักษณะ (attribute value) ซึ่งเป็นปัญหาที่มีลักษณะเป็นตรรกศาสตร์ประพจน์ (propositional logic) โดยใช้ค่าความจริงจากลักษณะสำคัญจัดเป็นอินพุตเวกเตอร์ (input vector) เพื่อป้อนให้แก่นิวรอลเน็ตเวิร์กเรียนรู้และทดสอบ นิวรอลเน็ตเวิร์กเป็นวิธีการเรียนรู้ของเครื่องแบบหนึ่งซึ่งใช้กันอย่างแพร่หลาย เนื่องจากนิวรอลเน็ตเวิร์กสามารถเรียนรู้เพื่อปรับค่าน้ำหนักของเส้นเชื่อมภายในโครงสร้าง ซึ่งเป็นการกำหนดความสำคัญให้แก่เส้นเชื่อมต่างๆ ดังนั้นหากนำลักษณะสำคัญมาป้อนเป็นอินพุตให้แก่นิวรอลเน็ตเวิร์กเมื่อผ่านกระบวนการเรียนรู้แล้ว นิวรอลเน็ตเวิร์กจะสามารถกำหนดได้ว่าลักษณะสำคัญข้อใดมีความสำคัญมากกว่าลักษณะสำคัญข้ออื่น สาเหตุสำคัญอีกประการหนึ่งที่เลือกใช้วิธีแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก คือ นิวรอลเน็ตเวิร์กสามารถจำแนกตัวอย่างที่มีลักษณะเป็นหลายกลุ่มได้ ดังนั้นด้วยการนำกระบวนการดึงลักษณะสำคัญและแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กมาใช้ร่วมกับกฎจากระบบไอแอลพี จะทำให้เราสามารถนำกฎที่ได้จากระบบไอแอลพีมาจำแนกตัวอย่าง ในกรณีที่ตัวอย่างนั้นไม่ตรงกับกฎข้อใดข้อหนึ่งพอดีได้

ผลการวิจัยที่ได้แสดงให้เห็นว่า การนำวิธีการดึงลักษณะสำคัญมาใช้ร่วมกับแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก ทำให้ประสิทธิภาพของการโปรแกรมตรรกะเชิงอุปนัยดีขึ้น สามารถจำแนกตัวอย่างที่ไม่ตรงพอดีกับกฎดั้งเดิมได้เป็นอย่างดี และผลการทดลองที่ได้แสดงให้เห็นถึงเปอร์เซ็นต์ความถูกต้องที่เพิ่มขึ้นของวิธีการที่นำเสนอ เมื่อเทียบกับระบบไอแอลพีอื่นที่มีอยู่ โดยได้ทำการทดลองกับชุดข้อมูลหลายชุด และวิธีการที่นำเสนอให้ผลที่ดีกว่าระบบไอแอลพีที่มีอยู่ทุกระบบในทุกชุดข้อมูล

5.5

สิ่งที่ดีเด่นในผลงานวิจัย

ผลงานวิจัยนี้มีจุดเด่นอยู่ที่ การนำแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก มาช่วยในการประมาณหากฎใกล้เคียงในกรณีที่ไม่มีกฎที่ตรงพอดีกับตัวอย่าง สามารถเพิ่มประสิทธิภาพของระบบไอแอลพี ได้อย่างดี เป็นงานวิจัยพื้นฐานซึ่งก่อให้เกิดความก้าวหน้าในงานวิจัยทางไอแอลพี และวิธีการนี้สามารถใช้ประยุกต์เพื่อเพิ่มประสิทธิภาพให้กับการจำแนกเว็บเพจเป็นหมวดหมู่ได้ นอกจากนั้นผู้วิจัยยังได้ นำแนวคิดเรื่องการใช้แบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กมาช่วยในการประมาณหากฎใกล้เคียงไปประยุกต์กับการเรียนรู้ของเครื่องอีกวิธีการหนึ่งคือ การเรียนรู้ต้นไม้ตัดสินใจ และพบว่าแนวคิดนี้ สามารถใช้งานได้อย่างกว้างขวางสามารถช่วยเพิ่มประสิทธิภาพของการเรียนรู้ต้นไม้ตัดสินใจได้อย่างดีด้วย ผลงานวิจัยนี้ได้ตีพิมพ์ในวารสารวิชาการนานาชาติหลายบทความ และยังได้รับการตีพิมพ์ในงานประชุมวิชาการนานาชาติอีกหลายบทความด้วยกัน บทความที่สำคัญๆ ได้แก่

- Boonserm Kijsirikul, Sukree Sinthupinyo and Kongsak Chongkasemwongse, "Approximate Match of Rules Using Backpropagation Neural Networks", Machine Learning Journal, Volume 44, Issue 3, pp.273-299, September, 2001.

- Nuanwan Soonthornphisaj, Boonserm Kijsirikul, "Iterative Cross-Training: An Algorithm for Web Page Categorization", Intelligent Data Analysis, Vol. 7(3) or 7(4) 2003 (to appear).

สถานที่ติดต่อ ผู้ช่วยศาสตราจารย์ ดร.บุญเสริม กิจศิริกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย
โทรศัพท์ ๐-๒๒๑๘-๖๙๗๖ โทรสาร ๐-๒๒๑๘-๖๙๕๕
E-mail: boonserm.k@chula.ac.th

5.6

Proceedings of the Second International Conference on Intelligent Technologies (InTech'2001), November 27 - 29, 2001 Edited by Tanprasert, Thitipong
Published by Faculty of Science and Technology, Assumption University, Bangkok, Thailand; ISBN: 974-615-063-5

An Evaluation of the Incremental Iterative Cross-Training Approach on Web Page Classification

Nuanwan Soonthornphisaj[1] and Boonserm Kijsirikul[2]

Machine Intelligence & Knowledge Discovery Laboratory
Department of Computer Engineering,
Faculty of Engineering,
Chulalongkorn University,
Phathumwan, Bangkok, 10330, Thailand.
E-mail: nuanwan[1]@chula.com, boonserm.k[2]@chula.ac.th

Abstract: The paper investigates an Iterative Cross-Training algorithm using the incremental labeling mode (I-ICT) with more challenging problems. The main concept of I-ICT is to iteratively train two sub-classifiers and classify unlabeled examples in crossing style. We conducted experiments on two Web page categorization problems in order to prove the robustness of the algorithm. We compare I-ICT against the supervised Naive Bayes classifier. The experimental results show that I-ICT still preserves its robustness performance in the Web page categorization tasks and has enough potential to be applied by a search engine.
Key words: Web page classification, Incremental Iterative Cross-Training

1. Introduction

The Internet is the biggest source of all kinds of information, which can be accessed by anyone through a search engine. As the number of web pages increases exponentially, it is more difficult to obtain the information rapidly. Therefore, an ideal search engine should have the most updated information of all web pages to provide the optimum search result for the user. One solution to this problem is to use a web robot to crawl the Internet and categorize the web pages beforehand. As we know, the Web page classification task is a tedious job and time-consuming task for human to read and analyze the category of the pages. Thus, we want it to be automatic and have high classification reliability.

The problem of text classification has been explored by many researchers with variety of learning algorithms [1,2,3,4,5,6,7]. When we give a sufficient set of labeled training examples, supervised learning is the most effective method for the classification. However, the construction of hand-labeled data must be done by human and thus this is a time-consuming process. Though it is costly to construct hand-labeled data, in some domains it is easy to obtain unlabeled ones, such as data in the Internet. Therefore, it is our interest to apply our algorithm to this problem in order to see some aspects and analyse its performance.

The Iterative Cross-Training (ICT) is a learning algorithm, which has been proven to be robust under the assumption that at least one classifier is supplied with domain knowledge. ICT has two labeling modes, which are batch-labeling and incremental-labeling. ICT was successfully applied to Thai Web page identification using the batch-labeling mode. The incremental-labeling mode was also investigated on the problem of Course/non-Course Web page classification. However, the impact of ICT in the exceptional case with more challenging problems is also an interesting aspect. Therefore, we employed the ICT in incremental-labeling mode (I-ICT) to do the Web page classification tasks.

We applied our method to two Web page classification problems. The first problem is the classification of Web pages into four categories, which are course, faculty, project and student homepage respectively. The second one is the classification of Web pages into four categories, which are car, motorcycle, jazz and astronomy. In order to make the explicit performance comparison of I-ICT, we also implement the supervised learning algorithm. The experimental results show that the performance of I-ICT is comparable to the classifier using the supervised learning algorithm.

The paper is organized as follows. Section 2 describes in detail about I-ICT and the classification mechanism of classifiers is also introduced. The concept of the supervised learning algorithm is explained in Section 3. Section 4 shows the experimental results. Discussion and conclusion will be given in Section 5 and 6, respectively.

2. Incremental Iterative Cross-Training

The architecture of ICT using incremental-labeling mode consists of two naive Bayes classifiers, each of which learns from different features of Web pages. The heading-based classifier considers words appearing in all headings of the Web page during the learning and classification process. The content-based classifier uses words appearing in the content of the Web page as the feature for the classification mechanism.

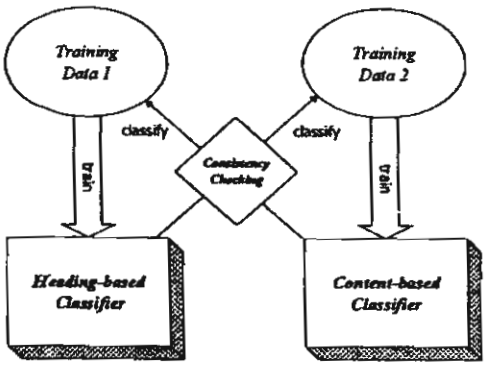


Figure 1 : The architecture of Incremental Iterative Cross-Training.

The process is started with a small number of initial labeled data. Each classifier estimates its parameters and uses the learned parameters to classify unlabeled data for the other as shown in Figure 1. The classification for unlabeled data is done in incremental way, i.e., the algorithm incrementally labels a small number of data. The training data is duplicated into two sets: *TrainingData1* for training the heading-based classifier and *TrainingData2* for training the content-based one. The heading-based classifier is trained by the labeled examples in *TrainingData1* to estimate its parameter set θ_h. With this current θ_h, the heading-based classifier will classify *TrainingData2* into predefined classes. Then the consistency checking process is performed to ask for the agreement from the content-based classifier.

The most confident p examples from each class will be labeled. The content-based classifier starts with parameter estimation by using labeled examples in *TrainingData1*. This process will be repeatedly done until all data are labeled.

Table 1. Incremental-ICT algorithm

Given:

- Two training sets *TrainingData1* of heading-based data and *TrainingData2* of content-based data (*TrainingData1* and *TrainingData2* both contain U labeled examples).
- Use labeled data in *TrainingData1* to estimate the parameter set θ_h of the heading-based classifier.
- Use labeled data in *TrainingData2* to estimate the parameter set θ_c of the content-based classifier.
- Loop until all data are labeled.
  - Use the content-based classifier with current θ_c to classify *TrainingData1* into categories.
    - Check consistency of the classification with the heading-based classifier. Label the class for the most confident p examples for each category.
    - Train the heading-based classifier by the labeled examples in *TrainingData1* to estimate the parameter set θ_h of the classifier.
  - Use the heading-based classifier with current θ_h to classify *TrainingData2* into categories.
    - Check consistency of the classification with the content-based classifier. Label the class for the most confident p examples for each category.
    - Train the content-based classifier by the labeled examples in *TrainingData2* to estimate the parameter set θ_c of the classifier.

The classification mechanisms of these two classifiers are the same, which use the naive Bayes algorithm. The algorithm is a well-known approach and is considered to be one of the most effective ways for text classification [10]. This algorithm employs *bag-of-words* to represent the document. The method is described below.

Given a set of class labels L = $\{l_1, l_2, \ldots, l_m\}$ and a document d of n words $(w_1, w_2, .., w_n)$, the most likely class label l^* estimated by naive Bayes is the one that maximizes $Pr(l_j|w_1, \ldots w_n)$:

$$l^* = \underset{l_j}{argmax}\ Pr(l_j|w_1, \ldots, w_n) \quad (1)$$

$$= \underset{l_j}{argmax}\ \frac{Pr(l_j)Pr(w_1, \ldots, w_n|l_j)}{Pr(w_1, \ldots, w_n)} \quad (2)$$

$$= \underset{l_j}{argmax}\ Pr(l_j)Pr(w_1, \ldots, w_n|l_j) \quad (3)$$

For our data set, L is the set of class labels, which are the categories of Web pages that we want the classifier to learn their concepts. $Pr(w_1, \ldots, w_n)$ in equation 2 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 $d = (w_1, w_2 \ldots, w_n)$, calculating the term $Pr(w_1, \ldots, w_n|l_j)$ 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. These assumptions are (1) the conditional independent assumption, i.e. the presence of each word is conditionally independent of all other words in the document given the class label, and (2) an assumption that the position of a word is unimportant, e.g. encountering the word "subject" at the beginning of a document is the same as encountering it at the end [10]. Equation 3 can be rewritten as:

$$l^* = \underset{l_j}{argmax}\ Pr(l_j) \prod_{i=1}^{n} Pr(w_i \mid l_j, w_1, \ldots, w_{i-1}) \quad (4)$$

$$= \underset{l_j}{argmax}\ Pr(l_j) \prod_{i=1}^{n} Pr(w_i \mid l_j) \quad (5)$$

The probabilities $Pr(l_j)$ and $Pr(w_i|l_j)$ are used as the parameter sets θ_h and θ_c of the classifiers, and are estimated from the 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 $Pr(w_i|l_j)$, of seeing word w_i given class label l_j, is estimated by the following equation:

$$Pr(w_i|l_j) = \frac{1 + N(w_i, l_j)}{T + N(l_j)} \quad (6)$$

Where $N(w_i, l_j)$ is the number of times word w_i appears in the training examples from class label l_j, $N(l_j)$ is the total number of words in the training set. T is the vocabulary size of the training set. Equation 6 employs Laplace smoothing (add one to all of word counts), to avoid assigning probability values of zero to words that do not occur in the training examples for a particular class.

To evaluate our method, we will compare it to the supervised naive Bayes classifier. The main idea of this classifier will be described in the next section.

3. Supervised Naive Bayes Classifier

The basic concept of *supervised learning* for building a classifier is that it requires a large set of examples with predefined classes. That means all of training data must be labeled. The classifier is then try to find some common properties of the different classes in order to make correct classification for unseen data. Thus, this kind of classifier needs a large number of labeled examples to correctly model the characteristic of the class during the learning process. Labeling must be done by human in order to train the classifier accurately. In our experiment, we employ the naive Bayes classifier as a supervised learning algorithm. The algorithm of the naive Bayes is the same as one described in Section 2, except that it is trained by hand-labeled data.

4. Experimental Results

In order to evaluate the impact of the I-ICT algorithm, we set up experiments on the problem of Web page categorization, and compare the performance of I-ICT to the supervised naive Bayes classifier. This section describes the data set, the setting of each classifier, and the results of the comparison on two data sets: (1) WebKb data set, and (2) WebClass data set.

4.1. The result on WebKb data set

The WebKb data set contains many Web pages related to the university domain. It is obtained via ftp from Carnegie Mellon University [8]. The data set consists of 981 Web pages collected from Computer Science department Web sites at four universities: Cornell, University of Washington, University of Wisconsin, and University of Texas. These Web pages have been hand-labeled into 4 categories, which are course homepage, faculty homepage, project homepage and student homepage.

In this data set, some categories are actually closely related which make the classification more difficult. A course home page gives information about the subject such as the course outline, the class schedule, reference books. A faculty homepage is an instructor homepage, which gives information about instructor's research, teaching course. A project homepage is actually a research homepage. A student homepage is a personal homepage of a student in the university.

Experimental Setting

We have 220 course Web pages, 147 faculty Web pages, 81 project Web pages and 533 student Web pages. Each sample is filtered to remove words that give no significance in predicting the class of the document. Words to be eliminated are auxiliary verbs, prepositions, pronouns, possessive pronouns, phone numbers, digit sequences, dates and special characters. Then, the word stemming process is applied to each sample by using Porter algorithm [11] in order to remove all suffixes and search for similar words based on the root word. Finally, we extract all headings appearing in each Web page to be the feature of the heading-based classifier. Therefore, each Web page can be viewed as a set of words appearing in the page's content and a set of words appearing in all headings.

The settings for the classifiers are as follow.

(1) For I-ICT, we randomly selected 30% of all samples from each category to be an initial labeled data. The training set (unlabeled data) consists of 30% of all samples and 40% of all samples were used as a test set. The parameter p was set to 1 for each class.

(2) For the supervised naive Bayes classifier, we supplied the algorithm with 60% of labeled data and 40% of all samples were used as a test set.

The experiments were conducted using 5-fold cross-validation in order to give each Web page a chance to be trained and tested equally.

The Results

Standard precision (P), recall (R), F_1-measure (F_1) are used to evaluate the performance of the classifiers. These measurements are defined as follows.

$$P = \frac{\text{no. of correctly predicted examples in the target class}}{\text{no. of predicted examples in the target class}} \quad (7)$$

$$R = \frac{\text{no. of correctly predicted examples in the target class}}{\text{no. of all examples in the target class}} \quad (8)$$

$$F1 = \frac{2PR}{P+R} \quad (9)$$

Table 2, 3 and 4 show the results of classifiers using the heading-based feature and the content-based feature respectively. In the table, "S-Bayes" stands for the supervised naive Bayes classifier. "I-ICT" is the naive Bayes classifier of the incremental Iterative Cross-Training classifier.

Table 2. The performance of classifiers using the Incremental Iterative Cross-Training algorithm.

I-ICT	Heading-based Classifier			Content-based Classifier		
category	P	R	F_1	P	R	F_1
course	89.38	95.00	92.10	88.94	94.55	91.66
faculty	54.22	84.32	66.00	47.02	82.32	59.85
project	79.84	66.54	72.59	58.03	60.59	59.28
student	93.53	78.27	85.22	94.81	71.31	81.40
average	79.24	81.03	80.13	72.20	77.19	74.61

Table 3. The performance of classifiers using the supervised Naive Bayes algorithm.

S-Bayes	Heading-based Classifier			Content-based Classifier		
category	P	R	F_1	P	R	F_1
course	89.32	94.54	91.86	86.98	92.27	89.55
faculty	56.85	88.43	69.21	46.70	83.65	59.94
project	85.01	77.50	81.08	58.16	65.00	61.39
student	94.32	78.11	85.45	95.85	69.89	80.84
average	81.38	84.65	81.90	71.92	77.70	72.93

Consider the average of F_1 measure, we found that the content-based classifier of I-ICT got 74.61% correctness which is higher than that of S-Bayes which got only 72.93%. That means, the content-based classifier of I-ICT can increase the correctness of S-Bayes 2.30%. However, the heading-based classifier of I-ICT got 80.13% which is less than the heading-based of S-Bayes only 2.16%.

In order to see the potential of I-ICT in acquiring the new label data during the training process, we set up another experiment of S-Bayes using exactly the same amount of labeled data as used by I-ICT. It means that the labeled data of S-Bayes was supplied with 30% of all examples. The results are shown in Table 4.

6.4

Table 4. The performance of classifiers using the supervised Naive Bayes algorithm (use the same amount of labeled data as I-ICT).

S-Bayes	Heading-based Classifier			Content-based Classifier		
category	P	R	F_1	P	R	F_1
course	86.45	92.27	89.27	86.86	91.36	89.05
faculty	56.02	85.03	67.55	49.07	79.49	60.68
project	80.22	70.22	74.89	56.47	59.34	57.87
student	92.17	77.34	84.11	93.47	73.74	82.44
average	78.72	81.22	78.95	71.47	75.98	72.51

We found that the performance measured by F_1 of heading-based and content-based classifiers of S-Bayes are decreased to 72.51% and 78.95%, respectively. Considering I-ICT which uses the same amount of labeled data, both classifiers' performance are higher than those of S-Bayes. This means, the learning mechanism of I-ICT could produce more true positive labeled data that enhance the correctness of both classifiers.

4.2 The result on WebClass data set

The WebClass data set was obtained from machine learning research group at Italy [13]. It consists of 192 Web pages corresponding to 4 categories, which are Astronomy, Jazz, Auto and Motorcycle. The first two categories are semantically distant while Auto and Motorcycle both concerning about vehicles are closely related.

Experimental Setting

The preprocessing step was done in the same way as in the WebKb data set.

The settings for classifiers are as follows.

(1) For I-ICT, we selected 33% from all examples to be initial labeled data. The training set consists of 33% and the rest 34% is a test set.

(2) For the supervised naive Bayes classifier, we select 66% from all examples to be labeled data. The test set is also 34% from all examples.

All experiments are conducted using 3-fold cross-validation. Table 5, 6 and 7 are the experimental results using the WebClass data set.

Considering the average performance measured by F_1 in Table 5 and 6, we found that the heading-based classifier of I-ICT got 74.61% which is higher than that of S-Bayes. However, the content-based classifier of I-ICT got 94.98% while S-Bayes got 95.03%. It means that the content-based classifier of I-ICT lost only 0.05% of accuracy compared to S-Bayes. Nevertheless, the performance of the heading- based classifier of S-Bays is less than that of I-ICT about 0.67%.

Table 5. The performance of classifiers using the Incremental Iterative Cross-Training algorithm.

I-ICT	Heading-based Classifier			Content-based Classifier		
Category	P	R	F_1	P	R	F_1
Astro	79.38	79.17	79.27	100.00	97.92	98.95
Auto	86.67	64.58	74.01	86.59	93.75	90.03
Jazz	57.41	91.67	70.60	100.00	100.00	100.00
Motor	86.77	52.08	65.09	94.12	87.50	90.69
average	77.56	71.88	74.61	95.18	94.79	94.98

Table 6. The performance of classifiers using the supervised Naive Bayes algorithm.

S-Bayes	Heading-based Classifier			Content-based Classifier		
Category	P	R	F_1	P	R	F_1
Astro	70.90	81.25	75.72	100.00	97.92	98.95
Auto	90.11	60.42	72.33	86.93	93.75	90.21
Jazz	63.49	91.67	75.02	100.00	100.00	100.00
Motor	81.48	54.17	65.07	94.12	87.50	90.69
average	76.50	71.88	74.11	95.26	94.79	95.03

Table 7. The performance of classifiers using the supervised Naive Bayes algorithm (use the same amount of labeled data as I-ICT).

S-Bayes	Heading-based Classifier			Content-based Classifier		
Category	P	R	F_1	P	R	F_1
Astro	66.79	70.83	68.75	100.00	97.92	98.95
Auto	90.56	54.17	67.79	84.87	91.67	88.14
Jazz	53.24	89.58	66.79	100.00	100.00	100.00
Motor	83.01	45.83	59.06	92.59	85.42	88.86
average	73.40	65.10	69.00	94.36	93.75	94.06

The potential of I-ICT in acquiring the new label data during the training process was also tested with the WebClass data set. Table 7 presents the result of the heading-based classifier and content-based classifier of S-Bayes using the same amount of labeled data as those of I-ICT. The average results measured by F_1 show that both classifiers of I-ICT outperform those of S-Bayes. The accuracy of the heading-based classifier of I-ICT is 8 13% higher than that of S-Bays. The content-based classifier of I-ICT is 0.98% higher than that of S-Bays. These experimental results show that using the same amount of labeled data, I-ICT has more potential than S-Bayes in classifying Web pages.

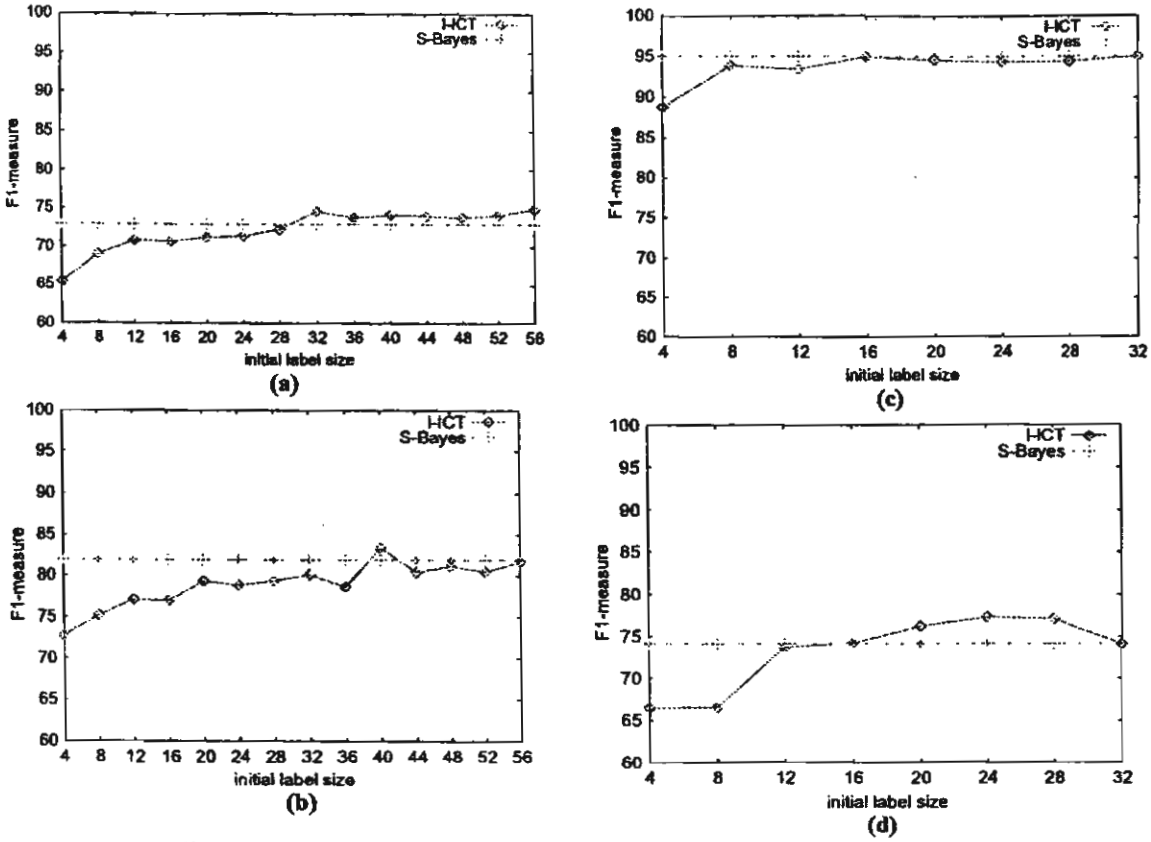


Figure 2. Performance of I-ICT on WebKb data set using content feature (a)
Performance of I-ICT on WebKb data set using heading feature (b)
Performance of I-ICT on WebClass data set using content feature (c)
Performance of I-ICT on WebClass data set using heading feature (d)

5. Discussion

The performance of I-ICT is competitive with S-Bayes in both data sets. Although, no domain knowledge is given, at least one classifier of I-ICT outperforms S-Bayes in both data sets. I-ICT has an advantage over S-Bayes because it needs less labeled examples than S-Bayes. In both data sets, I-ICT employs only 50% of labeled data used by S-Bayes, but it is able to produce a classifier that gives performance comparable to S-Bayes. It implies that under the restricted condition of the problem (the closely related of categories among classes and no domain knowledge is given), I-ICT still preserves its robustness on the classification task.

The performance of I-ICT using different initial label size was also investigated. As shown in Figure 2, we evaluated the performance of our algorithm based on F_1-measure in both data sets (WebKb and WebClass). The result shows that the number of initial label data plays an important role in the classification task. As the initial label size increases up to the optimum point, the performance of classifier is improved in all experiments. On WebKb data set, I-ICT (using content feature) outperforms S-Bays by using 32 initial label examples, whereas I-ICT using heading feature got the optimum performance by using 40 initial label examples. Considering the experiments on WebClass data set, we found that I-ICT (using content feature) could at least get the same performance as S-Bays using 16 initial label examples. For the I-ICT using heading feature, its performance is improved starting from 16 initial label examples.

The experimental results show the high tendency that I-ICT could perform well on multi-class problems. We believe that I-ICT has enough potential to deal with this kind of problems. We plan to build a powerful classifier with more informative feature sets. The special characteristics of Web pages are also challenging to the classification task. Further experiments on different data sets and with different parameters are planed for the near future in order to study whether the HTML structure of Web pages could provide a significant contribution on the Web page classification.

6. Conclusion

In this paper, we have demonstrated the concept of the I-ICT algorithm and conducted experiments on the Web page categorization problems. The I-ICT algorithm has been proven to be robust with more challenging problems. Our algorithm has an advantage over the supervised learning algorithm in the sense that the classifier needs only small amount of initial labeled data, whereas the supervised learning algorithm needs a huge number of labeled data.

Acknowledgement

This work has been partially supported by the Thailand Research Fund and National Electronics and Computer Technology Center (NECTEC) under the project number NT-B-06-4F-13-311.

References

[1] Attardi, G., Di Marco, D. Salvi., *Categorization by context.* On-line Proc. Of the 1st Int. Workshop on Innovative Internet Information Systems, 1998.

[2] Blum. A. and Mitchell, T. *Combining labeled and unlabeled data with Co-Training*, Proc. of the 11th Annual Conference on Computational Learning Theory, 1998.

[3] Cohen. W. and Singer, Y. *Context-sensitive learning methods for text categorization*, ACM Transactions on Information Systems, 17(2): 141-173, 1999.

[4] Joachims, T. *Text categorization with support vector machines: Learning with many relevant feature*, Proc. of 10th European Conference on Machine Learning, Springer Verlag, 1998.

[5] Jones, R., McCallum, A., Nigam, K. and Riloff, E., *Bootstrapping for text learning tasks*, IJCAI-99, Workshop on Text Mining: Foundations, Techniques and Applications, 52-63, 1999.

[6] Nigam, K., McCallum, A., Thrun, S. and Mitchell, T. *Text classification from labeled and unlabeled documents using EM.* Machine Learning, 39(2/3): 103-134, 2000.

[7] Apte, C., and Damerau, F. *Automated learning of decision rules for text categorization*, ACM TOES, 12(2): 233-251, 1994.

[8] Kijsirikul, B., Sasipongpairoege, P., Soonthornphisaj, N. and Meknavin, S. *Supervised and unsupervised learning algorithms for Thai Web pages identification*, Proc. of the Pacific Rim International Conference on Artificial Intelligence (PRICAI-2000), 690- 700. August 2000.

[9] David D. Lewis, Robert E. Schapire, James P. Callan and Ron Papka, *Training algorithms for linear text classifier*, Proc. of the 19th Annual Int. ACM SIGIR Conf. On Research and Development in Information Retrieval, 298-306, 1996.

[10] Mitchell, T. *Machine Learning*, McGraw-Hill. New York, 180-184, 1997.

[11] Porter, M. F.. *An algorithm for suffix stripping*, 130-137, 1980

[12] The World Wide Knowledge Base project, http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/webkb-data.gtar.gz, Carnegie Mellon University, U.S.A.

[13] TheWebClass project, University of bari, Italy. http: //www.di.uniba.it/ ~malerba/ software/ webclass / webclass.htm

[14] van Rijsbergen, C.J., *Information Retrieval*, Butterworths, London, 1979.

[15] Yang, Y., *An evaluation of statistical approaches to text categorization*, Information Retrieval Journal, 1999.

[16] Yang, Y., Pederson, J., *Feature selection in statistical learning of text categorization*, Proc. of the 14th International Conference on Machine Learning, 412-420, 1997.

Combining Neural Networks with Inductive Logic Programming for Predicting Unseen and Noisy Data

Sukree Sinthupinyo[1] and Boonserm Kijsirikul[2]
Department of Computer Engineering, Chulalongkorn University,
Phathumwan, Bangkok, 10330, Thailand
email: g40ssp[1], boonserm[2]@mind.cp.eng.chula.ac.th

Abstract: This paper presents a method for approximate match of first-order rules with unseen data. The method is useful especially in case of a multi-class or noisy domain where unseen data are often not covered by the rules. Our method employs a Backpropagation Neural Network for the approximation. To build the network, we propose a technique for selecting features from the rules to be used as inputs to the network. Our method has been evaluated on four domains of first-order learning problems. The experimental results show improvements of our method over the use of rules alone.
Key words: rule approximation, feature generation, inductive logic programming, backpropagation neural networks

1. Introduction

The advantages of *inductive logic programming (ILP)* [19, 13] are the expressive power of first-order logic hypotheses and the ability of employing background knowledge. ILP systems use background knowledge provided in form of first-order logic to generalize training examples, and produce rules that fit the training examples. However, when we apply an ILP system to a real-world domain, especially the domain where there are several classes of examples or the noisy domain, the produced rules may not cover or may not exactly match with the unseen data.

Consider for example the task of learning rules for classifying English uppercase characters. In this task, there are 26 classes of examples, i.e. 26 different English characters, and the real-world data usually contains noise such as noise due to the quality of the scanner. To classify English characters, we may use an ILP system to learn rules for each class. With exception of some systems which learn multi-class concepts [1, 2, 10, 12], most ILP systems work with two classes of examples (positive and negative) and construct a set of rules for the positive class. Any example not covered by the rules is classified as negative. If we want to employ these two-class systems to learn a multi-class concept, we could do this by first constructing a set of rules for the first class with its examples as positive and the other examples as negative, then constructing the sets of rules for other classes by the same process. The learned rules are then used to classify future data, and the rule that covers or exactly matches the data can be selected as the output. One major problem of this method is that some test data, especially noisy data, may not be covered by any rule. Thus the method is unable to determine the correct rule. A commonly used technique for solving this problem is to assign the majority class recorded from training data to the test data that is not exactly matched against any rule [4, 7].

We approach this problem directly by proposing a method to approximate the rule that provides the best match with the data. Here, we employ a Backpropagation Neural Network (BNN) for the approximation of ILP rules. The basic idea is that when there is no rule covering an example, we can make use of rules which *par-*

7.1

tially match with (partially cover) the example. Some of the partially matching rules may capture important *features (properties)*, and some may capture unimportant features of the examples. The best rule then should be the rule that matches many important features and does not necessarily match unimportant ones. The significance level of each feature is determined in terms of a weight that is trained by BNN. Our method can deal with a related problem when a test data is covered by multiple rules. We evaluate our method on four first-order datasets. The results show improvements of our method over the use of rules alone.

2. Approximate ILP Rules by a Backpropagation Neural Network

Several works have shown that combining neural networks with symbolic rules produced excellent performances [3, 11, 20]. In this paper, a multilayer feedforward neural network is employed to select the rule that closely matches with the input data. The algorithm for training the network used in our method is Backpropagation [17] that is widely applied to various classification problems.

The following subsections explain the methods for generating features, building network from features, and training the network.

2.1. Feature Generation

Our method is based on the idea that when there is no rule covering an example, we can make use of rules which *partially* cover (partially match with) the example, i.e. rules whose some literals are true for that example. The partially matching rule should not be neglected as it may capture some important properties or features of the example. The best rule should be the rule that matches many important features and does not necessarily match with unimportant ones. The significance level of each feature is determined in terms of a weight trained by BNN which will be described leter.

First, consider a first-order rule whose every literal in the body of the rule has only variables occurring in the head. For example, the following rule contains three literals in the body and all of them have no new variable.

```
mesh(A,11)← long(A),
            one_side_loaded(A),
            fixed(A).
```

Each of these literals is for checking a feature of an example. In such a rule, we will use each literal as a feature. We call this kind of feature *singleton feature*.

However, it is more difficult to determine what should be used as features when we consider first-order rules with new variables. A literal with new variables itself may not check for a specific property of the example, i.e. the literal alone may be meaningless without the presence of other literals which make use of the newly introduced variables. Most literals introducing new variables are for passing the introduced variables to other literals that may check a property or introduce other new variables again. Usually these newly introduced variables should end at a literal that checks for a property. The connection of these variables via the sequence of literals thus examines a feature of the example. Below we give an algorithm to select a sequence of literals for using as a feature. Our method of feature generation is based on the notion of closed chain.

Definition 1 *(Closed chain).* A sequence of some literals in the body of a rule is said to be a *closed chain* if every new variable not occurring in the head of the rule appears at least in two literals of the sequence and occurs at least once in a literal with variable(s) of the head or with variable(s) in one of the preceding literals.

Intuitively speaking, a new variable not occurring in the head of a rule is in a closed chains if after it is introduced by a literal it must be consumed by another literal. The closed chain does not allow a variable which occurs alone in two or more literals without being linked to existing variables. However, in some cases, some variables may not be in a closed chain.

Definition 2 *(Open chain).* A sequence of some literals in the body of a rule is said to be an *open chain* if there exists a new variable not occurring

in the head of the rule appears only once in a literal with variable(s) of the head or with variable(s) in one of the preceding literals.

The examples of closed and open chains are shown below.

Example 1 *(Closed chain)*. For the rule:

```
p(A,B)←  q1(A), q2(A,C), q3(C),
         q4(C,D), q5(D), q6(A,E,F),
         q7(E,G), q8(E,H), q9(E),
         q10(F,I), q11(I,B).
```

some closed chains are:

(i) `q2(A,C), q3(C)`
(ii) `q2(A,C), q4(C,D), q5(D)`
(iii) `q2(A,C), q3(C), q4(C,D), q5(D)`
(iv) `q6(A,E,F), q9(E), q10(F,I), q11(I,B)`

Example 2 *(Open chain)*. For the rule in Example 1, some open chains are:

(i) `q6(A,E,F), q7(E,G)`
(ii) `q6(A,E,F), q8(E,H)`
(iii) `q6(A,E,F), q7(E,G), q8(E,H)`
(iv) `q6(A,E,F), q7(E,G), q9(E)`
(v) `q6(A,E,F), q7(E,G), q8(E,H), q9(E)`

We then describe our method for generating features of a rule. The method is best understood by viewing a rule as a dependency graph. The root node of the graph is a set of variables occurring in the head of the rule. Each of the other nodes represents a set of new variables introduced by a literal, and an edge to the node represents the literal. The whole graph shows the dependency of variables. Figure 1 shows an example of dependency graph of the rule in Example 1.

Using the definitions of closed and open chains and viewing a rule as a dependency graph, we can now describe our algorithm for generating features as shown in Table 1.

The algorithm in Table 1 generates all closed chains that include variables at the root node of the graph. However, it does not generate all possible open chains. Open chains not generated are ones that are sub-chains of a closed chain features. This is because we consider that

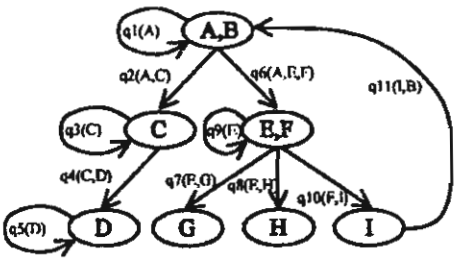


Figure 1: The dependency graph for the rule "p(A,B)←q1(A), q2(A,C), q3(C), q4(C,D), q5(D), q6(A,E,F), q7(E,G), q8(E,H), q9(E), q10(F,I), q11(I,B)."

Table 1: The algorithm for feature generation.

1. First find every edge beginning and ending at the root node and use it as a feature. Remove this kind of edges from the graph and do not consider the edges in the following steps.
   This type of feature is a *singleton feature* which introduces no new variable.
2. Find all possible closed chains starting from the root node, and use the sequence of literals along each of the chains as a feature.
   This type of feature is a *closed chain feature.*
3. For every leaf node that has no edge to others, find all possible paths that start from the root to that node.
   Use the sequence of literals along each of the paths as a feature.
   This type of feature is an *open chain feature.*
4. Find every possible combination of open chain features in Step 3 that has new variables (not occurring in the head) in common. If the combination is different from the existing open chain features, then add it to the feature set.

7.3

usually a newly introduced variable should be consumed by another literal that checks for a specific property of the example. Therefore, we first generate all closed chains if they exist; open chains which are sub-chains of a closed chain will not be generated. However, for some literal which introduces new variables, we may be unable to find a closed chain feature for the literal. In such a case, we generate an open chain feature that includes the literal. An example of feature generation is shown in Example 3.

Example 3 *(feature generation).* For the rule in Example 1, all possible features generated by our algorithm are:

Step 1. in Table 1
(i) `q1(A)`
Step 2. in Table 1
(ii) `q2(A,C), q3(C)`
(iii) `q2(A,C), q4(C,D), q5(D)`
(iv) `q2(A,C), q3(C), q4(C,D), q5(D)`
(v) `q6(A,E,F), q9(E), q10(F,I), q11(I,B)`
Step 3. in Table 1
(vi) `q6(A,E,F), q7(E,G)`
(vii) `q6(A,E,F), q8(E,H)`
(viii) `q6(A,E,F), q9(E), q7(E,G)`
(ix) `q6(A,E,F), q9(E), q8(E,H)`
Step 4. in Table 1
(x) `q6(A,E,F), q7(E,G), q8(E,H)`
(xi) `q6(A,E,F), q7(E,G), q8(E,H), q9(E)`

2.2. Building Network Structure from Features

Given a set of rules, we first generate features for each rule. The features of a rule are used as input units that are linked to one hidden unit which represents the rule. Therefore, the number of hidden units in the network is the same as the number of rules. Each class is represented by one output unit of the network. In two-class problems, there will be two output units, one for positive and the other for negative class. In multi-class problems, the number of output units is equal to the number of classes. The links from hidden units to output units are fully connected.

For example, consider the following rule set $\{C1, C2, C3, C4\}$.

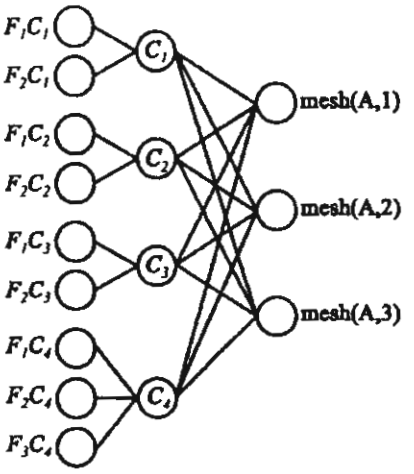


Figure 2: The structure of the neural network for the rule set $\{C_1, C_2, C_3, C_4\}$ in Section 2.2.

```
C1 : mesh(A,1) ← not_important(A),
                 not_loaded(A).
C2 : mesh(A,2) ← short(A),
                 opposite_l(B,A).
C3 : mesh(A,2) ← usual(A),
                 neighbour_yz_r(A,B),
                 cont_loaded(B).
C4 : mesh(A,3) ← short(A),
                 neighbour_zx_r(A,B),
                 opposite_r(A,C),
                 short(C).
```

The features for each rule are as follows, where F_iC_j is the i^{th} feature of the rule C_j.

F_1C_1: `not_important(A)`
F_2C_1: `not_loaded(A)`
F_1C_2: `short(A)`
F_2C_2: `opposite_l(B,A)`
F_1C_3: `usual(A)`
F_2C_3: `neighbour_yz_r(A,B), cont_loaded(B)`
F_1C_4: `short(A)`
F_2C_4: `opposite_r(A,C), short(C)`
F_3C_4: `neighbour_zx_r(A,B)`

Assume that there are only three classes, i.e. `mesh(A,1)`, `mesh(A,2)` and `mesh(A,3)`. Figure 2 shows the structure of the network for the above rules.

When new variables are considered, there can be many variable bindings for a rule that make different truth values for literals containing such variables. In our implementation, we use the binding that gives the maximum number of features whose truth values are true. The truth value of each feature is true if the truth values of all literals of the feature are true, otherwise the truth value of the feature is false.

2.3. Training the Network

The weights of the network are randomly initialized, and the final weights are obtained by standard backpropagation algorithm [17]. In our experiment, all units in the network use sigmoid activation function.

To train the network, each training example is evaluated with every rule and the truth values of features are determined. The features whose truth values are true are set to 1, whereas the features whose truth values are false are set to 0 for input units. The network is repetitively trained by using training examples until it converges or the number of training iterations exceeds the predefined threshold. After trained, the network can be used to classify unseen data. The unseen data is evaluated with features of each rule as in training process. The truth value of features are then fed into the network, and the output with highest value will be taken as the prediction.

3. Experiments & Results

We implemented a learning system, BANNAR (Backpropagation Artificial Neural Network for Approximating Rules) based on the above method. In the following experiments, we selected PROGOL [14] or GOLEM [16] for learning rules. Normally we used rules produced by PROGOL as the input to BANNAR. However in experiments on "Mutagenesis" and "King-Rook-King chess endgame" datasets described below, we did not successfully train PROGOL to produce a rule set. In those experiments, we employed GOLEM developed by the same research group of PROGOL. We then compared the results obtained by BANNAR with those of the rule set alone. To show the quality of the rule set used in our method, we also included the results obtained by the other two learning systems, i.e. 1BC and TILDE. 1BC is a first-order probabilistic learning system using naive bayes algorithm [8]. TILDE is a multi-class learning system that extends C4.5 to a first-order decision tree learner [2].

3.1. DataSets

Thai Character Recognition (TCR)

The dataset consists of 77 classes of examples, i.e. 77 different Thai characters.[1] The goal of this task is to learn rules for predicting the class for unseen data. In the training set, each character has 14 examples constructed from 14 sample images. The total number of training examples is 1,078. The noise were added into the original images, and the test data were constructed. The test set contains 2,143 test examples. This is a natural experimental setting as an unseen image usually contains noise when the learned rules or network are used by a character recognition software.

Each example is of the form `char(A,B,C,D,E,F)`. The information containing in `A,B,C,D,E,F` are *image features*[2] extracted by a pre-processing algorithm. These image features describe various properties of a character image such as the ratio of the width and the height of the character, the structure of lines and circles that form the character, the list of zones in the images that contain junctions of lines, etc. The background knowledge contains 55 predicates. See [9] for more details.

Finite Element Mesh Design (FEM)

The dataset for finite element mesh design [6], consists of 5 structures and has 13 classes (13 possible number of partitions for an edge in a structure). Each example is of the form `mesh(Edge,Number)` where `Number` indicates the number of partitions. The total number of examples is 278. The goal of finite element mesh design is to learn general rules describing how many elements should be used to model

[1] The dataset will be made available at http://www.mind.cp.eng.chula.ac.th.

[2] These are features describing the structure of the character images, such as the type of lines or circles contained in the images. These features should not be confused with the feature generation described in Section 2.1.

7.5

Table 2: The percent accuracies of BANNAR and the other systems on four datasets; TCR – Thai Character Recognition, FEM – Finite Element Mesh Design, MUTA – Mutagenesis, KRK – King-Rook-King Chess Endgame. Superscripts denote confidence levels for the difference in accuracy between BANNAR and the corresponding system, using a one-tailed paired t test: * is 90.0%, ** is 99.0%, * is 99.5%; no superscripts denote confidence levels that are below 90%. 3CV denotes the experiment that uses three-fold cross-validation. The accuracies of PROGOL or GOLEM are calculated by assigning the majority and negative class to uncovered examples in the case of multi-class and two-class problems, respectively.**

Dataset	# Train	# Test	# Classes	BANNAR	PROGOL or GOLEM	TILDE	1BC
TCR	1,076	2,143	77	94.40	72.00**	88.57**	77.23**
FEM	278	3CV	13	64.45	57.80*	58.02**	46.73**
MUTA	188	3CV	2	83.58	82.01	68.94	77.72*
KRK	10,000	3CV	2	99.83	99.76	69.67***	87.11***

each edge of a structure. The background knowledge consists of relations describing the properties of an edge (e.g. `short`, `not_loaded`), boundary conditions (e.g. `free`), loadings (e.g. `not_loaded`), and the relations describing the structure of the object (e.g. `neighbour`). See [5, 6] for more details.

Mutagenesis

The dataset for mutagenesis domain consists of 188 molecules, of which 125 are active and 63 are inactive. The goal of this problem is to predict the mutagenicity of the molecules, whether a molecule is active or inactive in terms of mutagenicity. This problem is a two-class learning problem. A molecule is described by listing its atoms `atom(AtomID,Element,Type,Charge)` and the bonds `bond(Atom1,Atom2,BondType)` between atoms. The background knowledge used in our experiment is the set $S2$ described in [18] that contains the definition of atom, bond, methyl groups, nitro groups, aromatic rings, hetero-aromatic rings, connected rings, ring length, and the three distinct topological ways to connect three benzene rings. See [18] for more details.

King-Rook-King Chess Endgame (KRK)

The last dataset used in our experiment is the KRK dataset provided by the Oxford university computing laboratory. [3] The task is to distinguish between illegal and legal board position [15]. The number of examples in the dataset is 10,000; 3,240 representing illegal KRK endgame positions (positive), the rest representing legal endgame positions (negative). Each example is of the form `illegal(WKf,WKr,WRf,WRr,BKf,BKr)`, where `(WKf,WKr)`, `(WRf,WRr)` and `(BKf,BKr)` are the positions (file,rank) of White King, White Rook and Black King, respectively. Each of these variable taking values from 0 to 7. Background knowledge contains two relations for comparing rank and file: `adj(X,Y)` and `lt(X,Y)` where `X`, `Y` are file or rank. Note that only in this dataset, for 1BC we used the background relations described in [8], such as `board2whiteking`, `board2blackking`,`board2whiterook`, `fileeq`, `rankeq`,`pos2rank`, etc. For the other datasets described above, all background relations given to all learning systems are the same.

3.2. Experimental Results & Discussions

We used three-fold cross-validation and averaged the results in all experiments except for the experiment on Thai character recognition dataset where training and test data are given. The experimental results are shown in Table 2.

The results show that the performance of PRO-

[3] http://www.comlab.ox.ac.uk/oucl/groups/machlearn/chess.html

Table 3: **Improvements of BANNAR over rules alone, reported according to covered and uncovered examples. The columns COVERED and UNCOVERED denote the numbers of examples covered and uncovered by the rules. Each cell denotes the number of examples correctly classified/the number of examples for that portion.**

Data Set	# Test	BANNAR		PROGOL or GOLEM (Majority or Negative Class)	
		COVERED	UNCOVERED	COVERED	UNCOVERED
TCR	2,143	1570/1611	453/532	1540/1611	3/532
FEM	278	145/200	34/78	146/200	14/78
MUTA	188	102/109	55/79	100/109	54/79
KRK	10,000	3352/3356	6638/6644	3350/3356	6633/6644

GOL or GOLEM is comparable to that of TILDE. It seems that PROGOL or GOLEM performed better than TILDE in two-class problems, whereas TILDE did better in multi-class problems. In the datasets tested in our experiment, 1BC did not perform well, compared to PROGOL or GOLEM. These results show the high quality of rules produced by PROGOL or GOLEM. Nevertheless, as shown in the table, BANNAR is still able to improve the accuracies of the rules, especially in the multi-class problems. Compared with the other learning systems, BANNAR performed best on all datasets. Moreoever, BANNAR significantly outperformed PROGOL or GOLEM, TILDE and 1BC on 2, 3 and 4 datasets, respectively. Note that in Mutagenesis domain, there are cases that multiple rules fire but there is no difficulty for BANNAR as it predicts the class which best matches the examples.

We further investigate these improvements. We want to see how well BANNAR correctly classify examples when they are not covered by the rules. Table 3 summarizes the results.

The results in Table 3 shows the ratio between the number of examples correctly classified and the number of examples for each portion. For example, 453/532 in the row TCR indicates that 532 examples were not covered by the rules, and 453 of them were correctly classified by BANNAR. 3/532 in the same row shows that 3 of 532 examples were correctly classified as we use majority class for predicting unseen data (or use negative class for two-class problems). This means that BANNAR correctly classified 450 more data in the case of uncovered examples. Similarly, 1570 of 1611 examples were correctly classified by BANNAR, whereas 1540 were correctly classifed by PROGOL or GOLEM; though the number of data covered by PROGOL or GOLEM was 1611, 1540 out of them were correct. The similar improvement can be seen on FEM dataset. The improvements were not significantly obtained on MUTA and KRK datasets which are two-class problems. These results show that BANNAR improved significantly on data which were not covered by the rules, especially in multi-class domains.

We further investigate if our method will help in two-class domains with noisy data. We choose the KRK dataset for doing an additional experiment. The next subsection describes the experiment and results.

3.3. An Additional Experiment on KRK Noisy Datasets

To study the effect of noise on two-class learning problems, we selected the KRK dataset. In the following experiment, three-fold cross-validation was used. The dataset was partitioned into three disjoint subsets. Each subset was used as a test set once, and the remaining subsets were used as the training set. Given training and test sets, 5%, 10% and 15% class noise was randomly added into the training set, and no noise was added into the test set. In our case, adding x% of noise means that class value was replaced with the wrong value in x out of 100 data. For example, 5% of noise means that 5% of data were randomly selected and the class values were replaced by the opposite value (from positive to

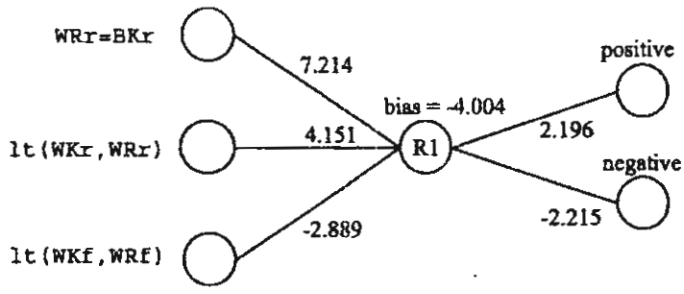


Figure 3: The portion of network for the rule "illegal(WKf,WKr,WRf,WRr,BKf,BKr) ← WRr=BKr, lt(WKr,WRr), lt(WKf,WRf)."

negative, and vice versa).

The average results of GOLEM [4] and BANNAR on noisy data are shown in Table 4. In the table, we also included the result on noise-free data for comparison.

Table 4: The percent accuracies of GOLEM and BANNAR with 5%, 10% and 15% noise added. The dataset contains 10,000 examples. The experiment was run using 3-fold cross-validataion. The number of rules is the average for 3-fold data.

Noise Levels	#Rule	BANNAR	GOLEM
0%	14.00	99.83	99.76
5%	49.67	98.09	92.27
10%	134.00	98.12	87.32
15%	150.00	94.33	82.51

As shown in the table, BANNAR significantly improved the accuracy of GOLEM when noise was added (the difference is statistically significant at a confidence level of 99.5% for 5% or 10% noise, and 97.5% for 15% noise). The table also shows that the number of rules increased when class noise was added. This means that the rules obtained by GOLEM were more specific and a large number of rules was needed to cover positive training data. These rules fit well only the training data, but they resulted in wrong classification for unseen data as shown by the decrease of the accuracy with increasing noise. The results of BANNAR show that the accuracy decreased much slower than that of GOLEM. This is because of the ability of BANNAR to give appropriate weights to features: higher weights to important features and lower weights to unimportant ones.

For instance, one of rules obtained when 5% noise was added is:

```
illegal(WKf,WKr,WRf,WRr,BKf,BKr) ←
        WRr=BKr,
        lt(WKr,WRr),
        lt(WKf,WRf).
```

The rule states that the position is illegal if (1) the ranks of the white rook and black king are the same, and (2) the rank of the white king is less than the rank of the white rook (thus the white king is not blocking the check), and (3) the file of the white king is less than (below) that of the white rook. Clearly, the feature (3) is not necessary if the feature (1) and (2) satisfy. This rule is an over-specific rule and it is likely that the rule overfits noisy data. The literal `lt(WKf,WRf)` should not be added to this rule, i.e., the rule will correctly classify more data if the literal is not included in the rule. When we employed BANNAR, the system finds the appropriate weight for each feature of the rule. In this case, each literal is selected as a feature. The unnecessary feature, i.e., `lt(WKf,WRf)`, was given a lower weight by BANNAR. Figure 3 shows the weights of literals of the rule. As shown in the figure, the weight of unuseful literal `lt(WKf,WRf)` is -2.889 and is dominated by the sum of weights of the others which is $7.214 + 4.151 = 11.365$. Therefore, if

[4] In the experiment, the number of pairs of examples to be considered for constructing rlggs is set to 50.

the first two features satisfy, this rule represented by the hidden unit $R1$ will give the positive output and makes a high chance of predicting the positive class.

The ability to give appropriate weights to feature is the advantage of BANNAR, because the unimportant feature will be received less attention in classifying unseen data. In this case, although the original rule is over-specific, but the obtained part of network is very useful to classify unseen data.

4. Conclusions

We have proposed a useful method that combines ILP and BNN for finding the first-order rules which best match the unseen data. Our method has been evaluated on four domains of first-order learning problems. The experimental results show that our method gives a significant improvements over the use of rules alone. The improved results come from the combination of ILP and BNN. ILP produces rules that accurately classify the training data, and BNN makes the rule more flexible for approximately matching with unseen or noisy data.

One direction for furthur reseach is to investigate more sophisticated method for evaluating the truth values of features, such as fuzzy logic. In the current work, when truth values of some of literals of a feature are false, the whole feature is assigned to be false. If we can assign more suitable value, it may increase the classification accuracy. Another interesting direction is to combine naive bayes classifier, like 1BC, with our method that generates features from rules, and use these features to learn probabilities for a naive bayes classifier.

Acknowledgement

This work is supported by the Thailand Research Fund and the Thailand-Japan Technology Transfer Project.

References

[1] Baroglio,C. and Botta,M. *Multiple Predicate Learning with RTL.*, Topic in Artificial Intelligence, LNAI 992, 1995.

[2] Blockeel,H. and Raedt,L.D., *Experiments with Top-Down Induction of Logical Decision Trees.* Technical Report CW247, Dept. of Computer Science, K.U.Leuven, 1997.

[3] Botta,M., Giordana,A. and Piola,R., *FONN: Combining First Order Logic with Connectionist Learning.* In Proceedings of the 14th International Conference on Machine Learning, 1997.

[4] Clark,P. and Niblett,T., *The CN2 Induction Algorithm.* Machine Learning Journal, 3(4), 261-283, 1989.

[5] Dolsak,B., Jezernik,A. and Bratko,I., *A knowledge base for finite element mesh design*, Artifical Intelligence in Engineering, 9, 19-27, 1994.

[6] Dolsak,B. and Muggleton,S., *The application of Inductive Logic Programming to Finite element mesh design.* In S. Muggleton, editor, Inductive logic programming, pp. 453-472. Academic Press, 1992.

[7] Dzeroski,S., Schulze-Kremer,S., Heidtke,K.R., Siems,K. and Wettschereck,D., *Applying ILP to Diterpene Structure Elucidation from 13C NMR Spectra.* In Proceedings of 6th International Workshop on Inductive Logic Programming, 1996.

[8] Flach,P. and Lachiche,N., *1BC: A First-Order Bayesian Classifier.* In Proceedings of the 9th International Workshop on Inductive Logic Programming, pp. 92-103, 1999.

[9] Kijsirikul,B. and Sinthupinyo,S., *Approximate ILP rules by Backpropagation Neural Network: A Result on Thai Character Recognition.* In Proceedings of the 9th International Workshop on Inductive Logic Programming, pp. 162-173, 1999.

[10] Laer,W.V., Dzeroski,S. and Raedt,L.D., *Multi-Class Problems and Discretization in ICL Extended Abstract.* In Proceedings of the Minet Familiarization Workshop on Data Mining with Inductive Logic Programming, pp. 53-60, 1996.

[11] Mahoney,J. and Mooney,R., *Comparing methods for refining certainty-factor rulebases.* In Proceedings of the 11th Inter-

national Workshop on Machine Learning, Ruthers University, NJ, 1994.

[12] Martin,L. and Vrain,C., *MULT_ICN: An Empirical Multiple Predicate Learner.* In Proceedings of the 5th International Workshop on Inductive Logic Programming, 1995.

[13] Muggleton,S., *Inductive logic programming,* New Generation Computing, 8(4), 295-318, 1991.

[14] Muggleton,S., *Inverse Entailment and PROGOL.* New Generation Computing, 3:245-286, 1995.

[15] Muggleton,S., Bain,M., Hayes-Michie,J. and Michie,D., *An experimental comparison of human and machine learning algorithms.* In Proceedings of the 6th International workshop on Machine Learning, Ithaca, NY, Morgan Kaufmann, 1989.

[16] Muggleton,S. and Feng,C., *Efficient induction of logic programs.* In Proceedings of the 1st Conference on Algorithmic Learning Theory, Tokyo, Ohmsha, 1990.

[17] Rumelhart,D.E., Hinton,G.E. and Williams,R.J., *Learning internal representations by error propagation.* In D. E. Rumelhart, & J. L. McClelland (Eds.), Parallel distributed processing (Vol 1), Cambridge, MA:MIT Press, 1986.

[18] Srinivasan,A., Muggleton,S., Sternberg,M.J.E. and King, R.D., *Theories for mutagenicity: A study in First-order and feature-based induction.* Artificial Intelligence, 85, 1996.

[19] Quinlan, J. R., *Learning logical definitions from relations.* Machine Learning, 5(3), 239-266, 1990.

[20] Towell,G.G. and Shavlik,J.W., *Knowledge-Based Artificial Neural Networks.* Artificial Intelligence, 70 (4):119-116, 1994.

Iterative Cross-Training: An Algorithm for Learning from Unlabeled Web Pages

Nuanwan Soonthornphisaj[1] and Boonserm Kijsirikul[2]
Machine Intelligence and Knowledge Discovery Laboratory
Department of Computer Engineering,
Chulalongkorn University,
Bangkok 10330, THAILAND
E-mail: nuanwan[1], boonserm[2]@mind.cp.eng.chula.ac.th

Abstract: The paper presents a learning method, called *Iterative Cross-Training (ICT)*, for classifying Web pages in two classification problems, i.e., (1) classification of Thai/non-Thai Web pages, and (2) classification of course/non-course home pages. Given domain knowledge or a small set of labeled data, our method combines two classifiers that are able to effectively use unlabeled examples to iteratively train each other. We compare ICT against the other learning methods: supervised word segmentation classifier, supervised naï ve Bayes classifier, and co-training-style classifier. The experimental results, on two classification problems, show that ICT gives better performance than those of the other classifiers. One of the advantages of ICT is that it needs only a small set of pre-labeled data or no pre-labeled data in the case that domain knowledge is available.
Key words: Iterative Cross-Training, Unlabeled data, Web page classification

1. Introduction

Given pre-labeled training data, supervised learning has been successfully applied to text classification [1,3,4,6,7,9,16]. 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 *active learning with committee* [10], text classification using EM [14], *co-training* algorithm [2].

This paper describes a new algorithm, called *Iterative Cross-Training (ICT)*, that effectively uses unlabeled data in the domain of Web page classification where unlabeled data is plentiful and easy to obtain. Our method combines two classifiers which iteratively train each other. 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 examples for the second classifier. The second classifier has no domain knowledge and learns its model from examples labeled by the first, and uses the current model to label training data for the first. This training process is iteratively repeated. With good interaction between two classifiers, the performance of the whole system is increasingly improved. In case that we have no domain knowledge, instead we supply the algorithm with a small number of labeled examples. One of the advantages of our method is that, as the method requires no labeled data or needs only a small number of data, it reduces human effort in labeling data and can be easily trained with a lot of unlabeled data.

We apply our method to two classification problems: (1) the classification of Web pages into Thai and non-Thai pages, and (2) the classification of Web pages into course and non-course pages which was introduced by Blum and Mitchell [2]. To evaluate the effectiveness of our method, we implement other classifiers to empirically compare with our method. The implementation is designed to explain, or at least give some answers to questions: "is 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 classifiers are: (1) supervised word segmentation classifier (*S-Word*), (2) supervised naï ve Bayes classifier (*S-Bayes*), (3) co-training-style classifier (*CoTraining*). Among these classifiers, S-Bayes or S-Word is single and supervised classifier. CoTraining and ICT are composed of two sub-classifiers and able to employ unlabeled data.

The experimental results show that ICT successfully and efficiently classify Web pages with high precision and recall. The overall performance, evaluated by F_1-*measure*, of ICT is better than those of the other methods tested in our experiments. The better performance of ICT than those of supervised ones (S-Bayes and

S-Word) demonstrates the successful use of unlabeled data. The results also show that the training technique of ICT is also an effective way as its performance is better than that of CoTraining which uses a different training technique.

The paper is organized as follows. Section 2 presents an overview of our system, and gives the details of our classifiers. Section 3 describes other learning methods used in our comparison. Section 4 describes the experimental results. Discussion and related work are given in Section 5. Finally, Section 6 concludes our work.

2. Iterative Cross-Training

This section presents the *Iterative Cross-Training (ICT)*. First we describe the architecture of our learning system, and then gives the details of two classifiers used in the system.

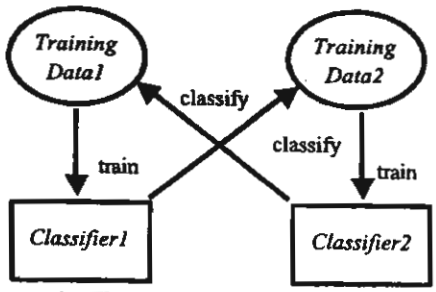


Figure 1: The architecture of Iterative Cross-Training. It is composed of two classifiers which use unlabeled data to iteratively train each other.

Figure 1 shows our learning system which learns to classify Web pages. The system is composed of two classifiers: *Classifier1* and *Classifier2*. Given domain knowledge or a small set of pre-labeled data, these two classifiers estimate their parameters from unlabeled data by receiving training from each other. Two training data sets, called *TrainingData1* and *TrainingData2* are duplicated from the unlabeled data provided by the user. Let θ_1 and θ_2 be sets of parameters of *Classifier1* and *Classifier2*, respectively. *TrainingData1* is used to train *Classifier1* to estimate its parameter set, and the *TrainingData2* is used to estimate the parameter set of *Classifier2*. 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 parameter set of *Classifier1* (θ_{1_0}) has property that it produces more true positive than wrong positive and more true negative than wrong negative examples for *TrainingData2*, the statistical information in correctly classified examples will be obtained.

Table 1: The training algorithm of Iterative Cross-Training.

Given:
- two sets *TrainingData1* and *TrainingData2* of unlabeled training examples

Initialize the parameter set of *Classifier1* to θ_{1_0}
$\theta_1 \leftarrow \theta_{1_0}$
Initialize the parameter set of *Classifier2* to θ_{2_0}
$\theta_2 \leftarrow \theta_{2_0}$

Loop until θ_1 does not change or the number of iterations exceeds a predefined value:

- If *labeling_mode*=BATCH Then
  - Use *Classifier1* with the current parameter set θ_1 to label all data in *TrainingData2* into positive examples and negative examples, and check consistency of the classification with *Classifier2* if necessary.

  Else * *labeling_mode*=INCREMENTAL *\
  - Use *Classifier1* with the current parameter set θ_1 to label the class for the most confident p positive unlabeled examples and most confident n negative unlabeled examples, and check consistency of the classification with *Classifier2* if necessary.
- Train *Classifier2* by using labeled examples in *TrainingData2* to estimate the parameter set θ_2 of *Classifier2*.
- If *labeling_mode*=BATCH Then
  - Use *Classifier2* with the current parameter set θ_2 to label all data in *TrainingData1* into positive examples and negative examples, and check consistency of the classification with *Classifier1* if necessary.

  Else * *labeling_mode*=INCREMENTAL *\
  - Use *Classifier2* with the current parameter set θ_2 to label the class for the most confident p positive unlabeled examples and most confident n negative unlabeled examples, and check consistency of the classification with *Classifier1* if necessary.
- Train *Classifier1* by the labeled examples in *TrainingData1* to estimate the parameter set θ_1 of *Classifier1*.

Using this information *Classifier2* should correctly classify more examples in *TrainingData1* that have similar characteristics. If the newly labeled *TrainingData1* can produce θ_1 better than θ_{1_0}, more reliable parameters of the whole system should be obtained after each iteration.

In the algorithm, first we initialize the parameter sets of *Classifier1* and *Classifier2*. This is done by training the classifiers with a small set of labeled

examples if they are available. If no labeled example provided to the system, the values of the parameters can be a pre-determined or randomly chosen ones. When a classifier labels data, it can ask for the confirmation from the other classifier to make decision about which class the example should be. If both classifiers agree with the same classifying result, that example will be labeled. The purpose of the consistency checking is for producing more reliable labeled data, but the checking will slow down the learning process.

As shown in Table 1, the algorithm has two labeling modes which are *batch-labeling* and *incremental-labeling*. The user must specify which labeling mode will be used in a particular problem. The difference between these two labeling modes is how the algorithm labels the data. In incremental-mode, the algorithm will incrementally produce a small set of new labeled examples at each round, but in batch-mode, the algorithm will label all examples and re-label them at each round. The batch-mode labeling tends to run fast, while the incremental-mode labeling tends to be more robust.

The following subsections describe the details of the classifiers.

2.1. Sub-Classifiers in ICT for the Classification of Thai/Non-Thai Web Pages

In the problem of classification of Thai/Non-Thai Web pages, our goal is to classify Web pages into Thai and non-Thai pages. This problem is of our interest because we want to build a Web robot that efficiently crawls the Web and retrieves only Thai pages for building a Thai search engine. In this problem, the first sub-classifier *Classifier1* is given some knowledge about the domain in form of dictionary and uses the dictionary for helping in determining whether a page is written in Thai or not. The algorithm used by *Classifier1* is word segmentation algorithm that will be described below. The second sub-classifier *Classifier2* is given no knowledge and uses the naï ve Bayes classifier.

(1) Word Segmentation Classifier (*Classifier1*)

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 hope 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 contains 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. [12].

Note that a string of Thai characters can usually be segmented in many possible ways because a word may be a substring of a longer word, and without a word delimiter it is difficult to find which segmentation is correct. Below we describe our method for word segmentation.

Given a Thai dictionary, 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.

$$\underset{w_1,\ldots,w_m}{\operatorname{argmin}} \sum_{i=1}^{m} \operatorname{cost}(w_i) \qquad (1)$$

where $\operatorname{cost}(w_i) = \eta 1$ if w_i is a word in the dictionary
$= \eta 2$ if w_i is a string not in the dictionary

In the following experiments, $\eta 1$ and $\eta 2$ are set to 1 and 2, respectively. As generating all possible segmentations and calculating their costs is very expensive, we employ dynamic programming technique to implement this calculation. Note that 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 it 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 word segmentation classifier (θ_i).

Having only the threshold of *WordRatio* (θ_i) as the parameter, we can find θ_{i0} which produces more true positive and true negative examples for *TrainingData2*. As describes 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 to see that any value of θ_{i0} will give more correctly classified pages than incorrectly ones (except for $\theta_{i0} = 0.0$ or $\theta_{i0} = 1.0$, 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_0, (e.g. 0.7, 0.8, 0.9) will produce more correctly classified pages. This is the case that is likely to be encountered in the real world. A low value of θ_0 is for the case that the number of Thai pages is larger than that of non-Thai pages.

A new θ can be estimated, after the naï ve Bayes classifier (*Classifier2*) 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. In case of $SP \geq LN$, the new θ is estimated as:

$$\theta = \frac{SP+LN}{2} \quad (2)$$

Now, consider the case of $SP<LN$. Let $V_1=SP$, $Vn=LN$, and $V_2,\ldots,V_{n-1}$ be the values between V_1 and V_n ($V_1 \leq V_2 \leq \cdots \leq V_{n-1} \leq V_n$). The new θ is estimated as:

$$\theta = \frac{V_{i^*} + V_{i^*+1}}{2} \quad (3)$$

$$V_{i^*} = \underset{V_i}{\text{argmin}} \text{ (no. of } V_j + \text{no.of } V_k)$$

Where V_k is a value of labeled positive example, V_j is a value of labeled negative example, and $V_1 \leq V_k \leq V_i$, $V_{i+1} \leq V_j \leq V_n$.

If *SP* is greater than *LN*, θ will completely discriminate the labeled positive from negative examples. Otherwise, θ will give the minimum errors of misclassified examples.

(2) Naï ve Bayes Classifier (*Classifier2*)

For text classification, naï ve Bayes is among the most commonly used and the most effective methods [13]. To represent text, the method usually employs *bag-of-words* representation. Instead of bag-of-words, we use the simpler *bag-of-characters* representation in the problem of classification of Thai/non-Thai pages. 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 identifying Thai 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 naï ve Bayes is the one that maximizes $\Pr(l_j|c_1,\ldots,c_n)$:

$$l^* = \underset{l_j}{\text{argmax}} \Pr(l_j \mid c_1,\ldots,c_n)$$

$$= \underset{l_j}{\text{argmax}} \frac{\Pr(l_j)\Pr(c_1,\ldots,c_n \mid l_j)}{\Pr(c_1,\ldots,c_n)} \quad (4)$$

$$= \underset{l_j}{\text{argmax}} \Pr(l_j)\Pr(c_1,\ldots,c_n/l_j) \quad (5)$$

In our case, L is the set of positive and negative class labels. The term $\Pr(c_1,\ldots,c_n)$ 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 $\Pr(c_1,c_2,\ldots,c_n \mid l_j)$ 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 naï ve Bayes are made [13]. 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. Clearly, these assumptions are violated in real-world data, but empirically naï ve Bayes has successfully been applied in various text classification problems [7,11,17].

Using the above assumptions, Equation 5 can be rewritten as:

$$l^* = \underset{l_j}{\text{argmax}} \Pr(l_j) \prod_{i=1}^{n} \Pr(c_i|l_j,c_1,\ldots,c_{i-1})$$

$$= \underset{l_j}{\text{argmax}} \Pr(l_j) \prod_{i=1}^{n} \Pr(c_i|l_j) \quad (6)$$

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_2 of our naï ve Bayes, and are estimated from the 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:

$$\Pr(c_i|l_j) = \frac{1+N(c_i,l_j)}{T+N(l_j)} \quad (7)$$

Where $N(c_i,l_j)$ is the number of times character c_i appears in training set from class label l, $N(l_j)$ is the total number of characters in the training set for class label l, and T is the total number of unique characters in the training set. Equation 7 employs Laplace smoothing (adding one to all the character counts for a class), to avoid assigning probability values of zero to characters that do not occur in the training data for a particular class.

2.2. Sub-Classifiers in ICT for the Classification of Course/Non-Course Home Pages

The problem of classification of Web pages into course/non-course pages is described in [2]. In this problem, each Web page contains two sets of features: (1) words appearing on the page, and (2) words appearing on the hyperlinks that link to that page. Therefore, each page can be viewed in two different ways, i.e., page-based features and hyperlink-based features. With these two feature sets, we construct two naï ve Bayes classifiers; the first one (*Classifier1* in Table 1) learns its model from hyperlink-features and the second one (*Classifier2*) learns from page-features. Both classifiers use naï ve Bayes algorithm which is the same algorithm described in the Section 2.1, except that for this problem the algorithm uses *bag-of-word* representation.

3. Other Classifiers Used in Comparison

In our experiment, we will compare Iterative Cross-Training with the following classifiers:

(1) supervised word segmentation classifier,
(2) supervised naï ve Bayes classifier, and
(3) co-training-style classifier.

Supervised word segmentation and supervise naï ve Bayes classifiers used in our comparison are the same as ones described in Section 2.1, except that they are trained by hand-labeled data. Co-training-style classifier is described as follows.

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. For example, a web page can be partitioned into the words occurring on that page, and the words occurring in hyperlinks that point to that page [2]. Either view of the example is assumed to be sufficient for learning. The algorithm consists of two sub-classifiers, each of which learns its parameter sets from each view of the example.

Based on this idea, we construct a co-training-style algorithm for our problems. The algorithm is shown in Table 2. The algorithm uses two sub-classifiers: *Classifier1* and *Classifier2*. These two classifiers are the same as ones of ICT:

(1) In the case of classification of Thai/non-Thai pages, we view each Web page as a set of words occurring in that page, and a set of characters occurring in the page. The word segmentation classifier (*Classifier1*) is employed to learn from the view of the word representation, and the naï ve Bayes classifier (*Classifier2*) is used for the character representation. The parameters θ_1 and θ_2 of *Classifier1* and *Classifier2* are estimated in the same way as described in Section 2.1.

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 the parameter set θ_1 of *Classifier1*.
- Use *LE* to estimate the parameter set θ_2 of *Classifier2*.
- Allow *Classifier1* with θ_1 to label p positive and n negative examples from *UE'*.
- Allow *Classifier2* with θ_2 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'*

(2) In the case of classification of course/non-course home pages, we view each Web page as words occurring on that page, and the words occurring in hyperlinks that point to that page. The page-based classifier, *Classifier1*, learns from words occurring on that page. The hyperlink-based classifier, *Classifier2*, learns from words occurring in the hyperlinks. For this problem, both *Classifier1* and *Classifier2* are naï ve Bayes classifiers

Our co-training-style algorithm is slightly different from the original one in that our algorithm will consume all data in *UE*. This is done to provide a fair comparison with the other methods. Allowing that all data to be consumed, there may be a case that the number of available positive or negative examples is not enough as required by the classifier. In such a case, the classifier is allowed to select examples with the other class.

4. Experimental Results

We conducted experiments to compare Interative Cross-Training (*ICT*) with the other classifiers described in the previous section: supervised word segmentation classifier (*S-Word*), supervised naï ve Bayes classifier (*S-Bayes*), and co-training-style classifier (*CoTraining*). This section describes the data set, the setting for each classifier, and the results of the comparison on two classification problems: (1) Thai/non-Thai page, and (2) course/non-course home page classification problems.

4.1. The Results on Thai/non-Thai Page Classification Problem

In this sub-section, we describe the data set and experimental setting for algorithms, and the results as follows.

Data Set & Experimental Setting

We collected the data set by starting from four Web pages: a Japanese Web page[1], two Thai Web pages[2], and an English web page[3]. From each of these four pages, a Web robot was 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). Note that HTML mark-up tags were removed before training and testing process. We used 3-fold cross validation in all experiments below for averaging the results.

The settings for the classifiers are as follows.

(1) For ICT, we ran the algorithm with both incremental and batch modes. Below we refer to incremental-mode ICT and batch-mode ICT as I-ICT and B-ICT, respectively. We used consistency checking for I-ICT and no consistency checking for B-ICT. No label data was given to BICT. The initial θ_0 was set to 0.7. For IICT, we gave 18 hand-labeled pages as initial labeled data for naï ve bayes classifier.

(2) For CoTraining, 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 $|UE|$, p, n and u to 1182, 3, 3 and 115, respectively.

The Results

To evaluate the performance of the classifiers, we use standard *precision(P)*, *recall(R)* and *F_1-measure*[4] *(F_1)* defined as follows:

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

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

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

[1] http://www.yahoo.co.jp

[2] http://www.sanook.com, http://www.pantip.com

[3] http://www.javasoft.com

[4] The F_1 measure has been introduced by van Rijsbergen [15] to combine recall and precision with an equal weight.

Table 3: The precision (%), recall (%) and F1-measure of the classifiers for the problem of Thai/non-Thai page classification.

Classifier	P (%)	R (%)	F_1
I-ICT(Word)	100.00	99.44	99.72
B-ICT(Word)	100.00	99.00	99.50
S-Bayes	100.00	99.00	99.50
B-ICT(Bayes)	100.00	98.89	99.44
I-ICT(Bayes)	99.55	99.33	99.44
CoTraining(Bayes)	100.00	98.89	99.44
S-Word	99.08	99.61	99.34
CoTraining(Word)	100.00	98.66	99.33

The results are shown in Table 3. In the table, "CoTraining(Bayes)" and "CoTraining(Word)" are the results of naï ve Bayes and word segmentation classifiers of *CoTraining*, respectively. "B-ICT(Bayes)" and "B-ICT(Word)" are for naï ve Bayes and word segmentation classifiers of ICT with the batch-mode while "I-ICT(Bayes)" and I-ICT(Word)" are those of the incremental-mode.

As shown in the table, I-ICT(Word) gave the best performance according to F_1-measure, followed by B-ICT(Word) which gave a comparable performance to SBayes. The performance of B-ICT(Bayes) was also comparable to that of CoTraining(Bayes) and I-ICT(Bayes). Compared to the other classifiers, S-Word and CoTraining(Word) did not perform well.

Compared to supervised classifiers, the performance of ICT was comparable to that of S-Bayes and quite better than that of S-Word. The results demonstrate that our system can effectively use unlabeled examples and the two modules succeed in training each other. The reason that I-ICT(Word) gave better performance than B-ICT(Word) comes from the consistency checking step during the classification processes. Though we did not include the details of running time of all classifiers, from the experiments we found that B-ICT ran much faster than I-ICT and CoTraining.

4.2. The Results on Course/non-Course Home Page Classification Problem

Below we describe the data set and experimental setting, and the results on the course/non-course page classification problem.

Data Set & Experimental Setting

The data for our experiment is obtained via ftp from

Carnegie Mellon University[5]. It consists of 1,051 Web pages collected from Computer Science department Web sites at four universities: Cornell, University of Washington, University of Wisconsin, and University of Texas. These Web pages have been hand-labeled into two categories. We consider the category "course home page" as the positive class and the other as the negative class. In this dataset, 22% of the Web pages are course home pages.

Each example is filtered to remove words which give no significance in predicting the class of the document. Words to be eliminated are auxiliary verbs, prepositions, pronouns, possessive pronouns, phone numbers, digit sequences, dates and special characters. We have 230 course Web pages and 821 non-course Web pages. Each Web page has two views, page-based and hyperlink-based, respectively. The training set contains 172 course Web pages and 616 non-course Web pages. Three positive examples and nine negative examples were randomly selected from the training dataset to be the initial labeled data. Therefore, each data set contains 12 initial labeled examples, 776 unlabeled training examples and 263 test examples. We then used 3-fold cross-validation for averaging the results.

The settings for the classifiers are as follows.
(1) For ICT, we ran the algorithm with both incremental and batch modes using consistency checking. As we have no domain knowledge to provide to the classifier for this problem, we gave 3 positive and 9 negative examples as initial labeled data for ICT. The parameters p and n in Table 1 were set to 1 and 3, respectively.

(2) For CoTraining, the values of the parameters of the classifier (in Table 2) were set in the same way as in [2]. As CoTraining requires a small set of pre-classified training data, we gave the algorithm with 3 positive and 9 negative examples. In our experiment, we set the values of $|UE|$, p, n and u to 776, 1, 3 and 75, respectively.

The Results

The experimental results are shown in Table 4. In Table 4, I-ICT(Page) and I-ICT(Hyperlink) stand for the page-based and hyperlink-based naï ve Bayes classifiers of I-ICT, respectively, and B-ICT(Page) and B-ICT(Hyperlink) are those of B-ICT. CoTraining(Page) and CoTraining (Hyperlink) are page-based and hyperlink-based naï ve Bayes classifiers of CoTraining algorithm, respectively. S-Bayes(Page) and S-Bayes

[5] The Word Wide Knowledge Base (web-kb) project, [http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-51/www/co-training/data/course-co-train-data.tar.gz], Carnegie Mellon University

Table 4: The precision (%), recall (%) and F-measure of the classifiers for the problem of course/non-course page classification.

Classifier	P (%)	R (%)	F_1
I-ICT(Page)	94.04	80.46	86.72
S-Bayes (Page)	77.48	94.35	85.09
S-Bayes(Hyperlink)	87.81	62.17	72.80
I-ICT(Hyperlink)	67.54	72.41	69.89
CoTraining(Hyperlink)	62.41	59.19	60.75
CoTraining(Page)	91.91	34.49	50.15
B-ICT(Page)	67.70	39.76	50.10
B-ICT(Hyperlink)	62.11	34.08	44.01

(Hyperlink) are supervised naï veBayes classifiers, which classify Web pages based on words in Web pages and words in hyperlinks, respectively.

As shown in the table, I-ICT(Page) gave the best performance followed by S-Bayes(Page), S-Bayes(hyperlink), I-ICT (Hyperlink), CoTraining (Hyperlink) and CoTraining(Page). The performance of B-ICT's were lower than the others. Compared to the performance of B-ICT on Section 4.1, the results of B-ICT on this problem were not good. This is due to the fact that unlike B-ICT on Section 4.1 which was given knowledge in form of dictionary, B-ICT on this problem had no knowledge about the domain. In this problem, B-ICT received only a small set of labeled examples for building its initial parameter set. As shown by the results, this initial parameter set did not contain enough statistical information for labeling the whole examples in batch-mode. However, when we ran the algorithm with incremental-mode, with the help of consistency checking, I-ICT incrementally added a small set of examples on each round, and gave an improved results over B-ICT.

The reason that I-ICT(Page) gave better performance compared to S-Bayes is because I-ICT(Page) cooperated with I-ICT(Hyperlink) while S-Bays used single classifier. The performance of I-ICT(Hyperlink) was not good as that of I-ICT(Page). This is because hyperlinks contain fewer words and thus are less capable of building accurate classifier. The training technique of I-ICT is also an effective way as its performance was better than that of Co-Training which uses a different training technique.

5. Discussion and Related Work

We have applied ICT on two classification problems. The problem of Thai/non-Thai page classification is simpler than the problem of

course/non-course home page classification. This can be seen by the performance of all classifiers which decreased on the second problem. For a difficult problem, incremental-mode ICT seems to be more suitable than batch-mode ICT. Batch-mode ICT has an advantage that it run fast, and it is suitable for the problem where we can provide domain knowledge.

Though the performance of our method is comparable or better than the other classifiers, the precision and recall on the problem of course/non-course page classification are still not high. This may be due to the simple model of the classifiers, i.e., naï ve Bayes classifiers. We plan to construct some domain knowledge for giving to the classifier and employs more powerful classifiers to test in this problem in the near future.

Our technique is related to Expectation-Maximization algorithm [5]. EM algorithm is an effective method for dealing with missing values in data, and has successfully been applied to text classification [14]. Nigam, et al. [14] have demonstrated that the accuracy of classifiers can be improved by using EM to augment a small number of labeled data with a large set of unlabeled data.

Meta-bootstrapping is another unsupervised algorithm for learning from unlabeled data [8]. Like our method, the algorithm is composed of two sub-learning algorithms. However, the training process of meta-bootstrapping and the way of using data are different from our method. This algorithm is multi-level algorithm and is very useful, especially in the complex domain where sub-learning algorithms alone could not produce enough good results. We also plan to study this kind of multi-level algorithm for using with our method.

6. 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 sub-classifiers that iteratively train each other. With no pre-labeled or a small set of pre-labeled examples, our method gives high precision and recall on classifying Web pages. The performance of our method is competitive with those of supervised ones, which demonstrates the successful use of unlabeled data of our method.

Acknowledgement

This paper is supported by Thailand Research Fund and National Electronics and Computer Technology Center.

References

[1] Apte, C, & Damerau, F., Automated Learning of Decision Rules for Text Categorization, *ACM TOIS* 12 (2): 233-251, 1994.

[2] Blum, A., & Mitchell, T., Combining labeled and unlabeled data with co-training, *Proceeding of the Eleventh Annual Conference on Computational Learning Theory*, 1998.

[3] Cohen, W. W., Fast effective rule induction, *Proceedings of Twelfth International Conference on Machine Learning*, Morgan Kaufmann, 1995.

[4] Cohen, W. W., & Singer, Y., Context-sensitive learning methods for text categorization, *ACM Transactions on Information Systems*, 17 (2): 141–173, 1998.

[5] Dempster, A. P., Laird, N. M., & Rubin D. B., Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society*, Series B, 39 (1): 1-38, 1977.

[6] Joachims, J., A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. *Proceedings of the Fourteenth International Conference on Machine Learning* 143-151, Morgan Kaufmann, 1997.

[7] Joachims, T., Text categorization with support vector machines: Learning with many relevant features, *Proceedings of the Tenth European Conference on Machine Learning*, Springer Verlag, 1998.

[8] Jones, R., McCallum, A., Nigam, K., & Riloff, E., Bootstrapping for text learning tasks, *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, 52-63, 1999.

[9] Lewis, D., Naive (Bayes) at forty: The independence assumption in information retrieval, *Proceedings of the Tenth European Conference on Machine Learning*, 1998.

[10] Liere, R., & Tadepalli, P., Active learning with committees for text categorization, *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 591-596, 1997.

[11] McCallum, A., Rosenfeld, R., Mitchell, T. & Nigam, A., Improving text classification by shrinkage in a hierarchy of classes, *Proceedings of the Fifteenth International Conference on Machine Learning*, 350-358, Morgan Kaufmann, 1998.

[12] Meknavin, S., Charoenpornsawat, P., & Kijsirikul, B., Feature-based Thai word segmentation, *Proceeding of Natural Language Processing Pacific Rim Symposium '97*, 1997.

[13] Mitchell, T., *Machine Learning*, 180-184, McGraw-Hill. New York, 1997.

[14] Nigam, K., McCallum, A., Thrun, S., & Mitchell, T., Text classification from labeled and unlabeled documents using EM, *Machine Learning*, 2000 (to appear).

[15] van Rijsbergen, C. J., *Information Retrieval*, Butterworths, London, 1979.

[16] Yang, Y., An evaluation of statistical approaches to text categorization, *Information Retrieval Journal*, 1999.

[17] Yang, Y., & Pederson, J., *Feature selection in statistical learning of text categorization*, Proceedings of the Fourteenth International Conference on Machine Learning, 412-420, Morgan Kaufmann, 1997.

8.9

International Conference on Information Technology Interface
ITI-2002, 2002

Web Page Categorization using Hierarchical Headings Structure

Nuanwan Soonthornphisaj[12], Pisit Chartbanchachai[2], Thanapol Pratheeptham[2] and Boonserm Kijsirikul[2]

[1]*Department of Computer Science*
Faculty of Science
Kasetsart University
Phaholyothin Rd. Bangkok, 10900, Thailand
E-mail: fscinws@ku.ac.th

[2]*Machine Intelligence and Knowledge Discovery Laboratory*
Department of Computer Engineering
Chulalongkorn University,
Phathumwan, Bangkok, 10330, Thailand.
E-Mail: boonserm.k@chula.ac.th

Abstract. *The goal of Web page categorization is to classify the Web documents into a certain number of predefined categories. The previous works in this area employed a large number of labeled training documents for supervised learning. The problem is that, it is difficult to create the labeled training documents. While it is easy to collect the unlabeled documents, it is not so easy to manually categorize them for creating training documents. Therefore, a new machine learning algorithm should be investigated to overcome these difficulties. We proposed a new algorithm called Iterative Cross-Training* (ICT). *The paper also present a new feature set which is the hierarchical structure of headings appearing in the Web page to enhance the classification performance. We found that the hierarchical structure of headings has a high impact and could enhance the classification performance.*

Keywords. Machine Learning, Web page categorization, feature sets.

1. Introduction

The availability of large, heterogeneous repositories of Web pages is increasing rapidly. There are billions pages accessible on the Internet with 1.5 million pages being added daily [4]. A user searching for documents within a specific category using a general purposed search engine might have a difficult time finding valuable documents. Search engines Web sites, such as a Yahoo[1], Google[2] etc., organize their Web resources in category-specific style. These Web sites currently use human experts to categorize the documents. However, the growth of Web pages nowadays is exponentially increased. It is difficult to keep updating and maintaining the index of billions Web pages. To improve category specific search, we need a well-trained classifier with a high ability to recognize Web pages of a specified category.

Many traditional machine learning techniques were investigated and applied to the Web pages categorization problem. The algorithm called bootstrapping was investigated in the domain of text learning by Rosie Jones [2]. This algorithm needs knowledge about the classes of document, which is provided in the form of a few keywords per class and a class hierarchy. The algorithm proceeds by using the keywords to generate preliminary labels for some documents by term matching. Then these labels, the hierarchy and all of unlabeled document become the input to a bootstrapping algorithm. The bootstrapping algorithm combines 2 techniques, which are the hierarchy shrinkage and Expectation–Maximization (EM) with unlabeled data. They tested the algorithm with the topic identification of computer science research papers. The experimental result shows that the algorithm could get 66% of correctness.

The Co-Training algorithm was first introduced by [1]. The concept of the algorithm

[1] http://www.yahoo.com
[2] http://www.google.com

is based on the boosting technique. That means, the algorithm learn from a small initial labeled data then it will incrementally classify unlabeled data into categories. The basic assumption of Co-Training is that, the instance distribution is compatible with the target function. It requires that, for most examples, the target functions over each feature set predict the same label. For example, in the web page domain, the class of the instance should be identifiable using either the hyperlink text or the page text alone. The second assumption is that the features in one set of an instance are conditionally independent of the features in the second set, given the class of the instance. This assumes that the words on a web page are not related to the words on its incoming hyperlinks.

We applied our method to two Web page classification problems. The first problem is the classification of Web pages into four categories, which are course, faculty, project and student homepage respectively. The second one is the classification of Web pages in the pharmaceutical domain. In order to make the explicit performance comparison of I-ICT, we also implement the supervised learning algorithm and Co-Training algorithm. The experimental results show that the performance of I-ICT is comparable to the classifier using the supervised learning algorithm.

The paper is organized as follows. Section 2 describes in detail about the proposed feature sets used in our experiments. Section 3 introduces the concept of I-ICT and the classification mechanism of classifiers. The Co-Training algorithm will be explain in Section 4.The concept of the supervised learning algorithm is explained in Section 5. Section 6 shows the experimental results. Discussion and conclusion will be given in Section 7 and 8, respectively.

2. Feature Sets

For the classification problem, the classifier's performance usually depends on the classification mechanism with the support of feature sets. The appropriate feature sets will help the classifier to enhance its classification correctness. Therefore we try to investigate the possible feature sets to see their contribution on the precision and recall of the classifier. Feature sets that we study are as follows

2.1 Content

The content of a Web page provides information to the user in detail. It is considered to be the main resource for the text categorization problem, whether it is done by human expert or by an automatic classifier. Therefore, we extract all words in the content to be the feature set in our experiments.

2.3 Hierarchical Headings

The heading phrase normally represents the main idea of the following content. Considering a Web page, we found that, it is normally organized into a hierarchical style; the main heading is usually followed by the sub-headings. Therefore, this structure should somehow represent the concept of the following content. We use this opportunity to extract all headings in the page and assign weight for words appearing in the heading related to the hierarchical structure. The weight assignments are shown in Table 1.

Note that, Table 1 also include some html tags that are used to make the different style of the text, such as the <b> tag which use to make the bold style, <i> is used to make an italic style.

Table 1. The weight assignment for headings appeared in the Web page.

Html Tag	weight
<title>	10
<META NAME="description"	10
<META NAME="keyword"	10
<META NAME="rating"	10
<h1>	9
<h2>	8
<h3>	7
<h4>	6
<st>	5
	4
<blockquote>	3
<a href=''>	2
<a>	2
	2
<th>	2
<b>	2
<u>	2
<i>	2

3.Incremental Iterative Cross-Training

The architecture of our learning algorithm consists of two naive Bayes classifiers, each of which learns from different features of a Web page. For the ease of explanation, we will use the concrete example of feature sets which are words appeared on the heading (heading-based) and words on the page (content-based). Starting with a small number of labeled data, each classifier estimates its parameters and uses the learned parameters to classify unlabeled data for the other as shown in Figure 1. The classification for unlabeled data is done in incremental way, i.e., the algorithm incrementally labels a small number of data. The training data is duplicated into two sets: *TrainingData1* for training the heading-based classifier and *TrainingData2* for training the content-based one. The concept of our algorithm is that if we could obtain reliable statistical information from the first classifier, it should be useful in classifying training data for the second classifier. After receiving training from each other, the parameters of the classifiers should be more reliable every iteration.

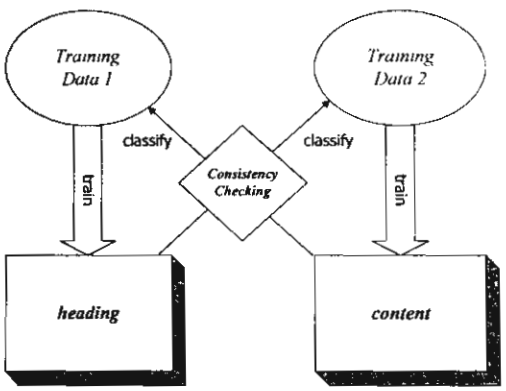


Figure 1. The architecture of Incremental Iterative-Cross Training algorithm.

Given a set of class labels L = $\{l_1, l_2, \ldots, l_m\}$ and a document d of n words $(w_1, w_2, .., w_n)$, the most likely class label l^* estimated by naive Bayes is the one that maximizes $Pr(l_j|w_1, \ldots w_n)$:

$$l^* = \underset{l_j}{argmax}\ Pr(l_j|w_1, \ldots, w_n) \quad (1)$$

$$= \underset{l_j}{argmax}\ \frac{Pr(l_j)Pr(w_1, \ldots, w_n|l_j)}{Pr(w_1, \ldots, w_n)} \quad (2)$$

$$= \underset{l_j}{argmax}\ Pr(l_j)Pr(w_1, \ldots, w_n|l_j) \quad (3)$$

Table 2. Incremental - ICT algorithm

Given:

- Two training sets *TrainingData1* of heading-based data and *TrainingData2* of content-based data (*TrainingData1* and *TrainingData2* both contain U labeled examples).
- Use labeled data in *TrainingData1* to estimate the parameter set θ_h of the heading-based classifier.
- Use labeled data in *TrainingData2* to estimate the parameter set θ_c of the content-based classifier.
- Loop until all data are labeled.
  - Use the content-based classifier with current θ_c to classify *TrainingData1* into categories.
    - Check consistency of the classification with the heading-based classifier. Label the class for the most confident p examples for each category.
    - Train the heading-based classifier by the labeled examples in *TrainingData1* to estimate the parameter set θ_h of the classifier.
  - Use the heading-based classifier with current θ_h to classify *TrainingData2* into categories.
    - Check consistency of the classification with the content-based classifier. Label the class for the most confident p examples for each category.
- Train the content-based classifier by the labeled examples in *TrainingData2* to estimate the parameter set θ_c of the classifier.

For our data set, L is the set of class labels $Pr(w_1, \ldots, w_n)$ in equation 2 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 $d = (w_1, w_2, \ldots, w_n)$, calculating the term $Pr(w_1, \ldots, w_n|l_j)$ 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. These assumptions are (1) the conditional independent assumption, i.e. the presence of each word is conditionally independent of all other words in the document given the class label, and (2) an assumption that the position of a word is unimportant, e.g. encountering the word "subject" at the beginning of a document is the

same as encountering it at the end (Mitchell, 1997). Equation 3 can be rewritten as:

$$l^* = \underset{l_j}{argmax}\ Pr(l_j) \prod_{i=1}^{n} Pr(w_i \mid l_j, w_1, \ldots, w_{i-1}) \quad (4)$$

$$= \underset{l_j}{argmax}\ Pr(l_j) \prod_{i=1}^{n} Pr(w_i \mid l_j) \quad (5)$$

The probabilities $Pr(l_j)$ and $Pr(w_i|l_j)$ are used as the parameter sets θ_h and θ_c, and are estimated from the 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 $Pr(w_i|l_j)$, of seeing word w_i given class label l_j, is estimated by the following equation:

$$Pr(w_i|l_j) = \frac{1 + N(w_i, l_j)}{T + N(l_j)} \quad (6)$$

Where $N(w_i, l_j)$ is the number of times word w_i appears in the training examples from class label l_j, $N(l_j)$ is the total number of unique word in the training set. T is the number of class. Equation 6 employs Laplace smoothing (add one to all of word counts), to avoid assigning probability values of zero to words that do not occur in the training examples for a particular class.

4. The Co-Training Algorithm

The Co-Training algorithm explicitly uses the split of the features when learning from labeled and unlabeled data. Its approach is to build the naive Bayes classifier for each of the distinct feature sets. Each classifier is initialized using a few labeled documents. Then every round of Co-Training, each classifier chooses the most confident p positive and n negative labeled examples to add to the labeled set of documents. The documents selected are those that have the highest posterior class probability, $Pr(l_j|d)$. Then, each classifier rebuilds from the augmented labeled set and the process repeats [1].

Table 3: The Co-Training 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 while there exist documents without class labels:

- Use *LE* to estimate θ_h of the hyperlink-based classifier using the hyperlink portion of each document.
- Use *LE* to estimate θ_c of the content-based classifier using the page portion of each document.
- Allow the hyperlink-based classifier with current θ_h to label p positive and n negative examples from *UE'*.
- Allow the content-based classifier with current θ_c 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'*.

5. Supervised Naive Bayes Algorithm

The basic concept of supervised learning for building a classifier is that it requires a set of examples with predefined classes.The classifier is then try to find some common properties of the different classes in order to make correct classification for unseen data. Thus, this kind of classifiers need a large number of labeled examples to correctly model the characteristic of the class during learning process. Labeling must be done by human to train the classifier accurately. In our experiment, we employ the naive Bayes classifier as a supervised learning algorithm. The algorithm of the naive Bayes is the same as one described in Section 3, except that it is trained by hand-labeled data.

6. Experimental Results

In order to test the robustness of the incremental-ICT algorithm and to investigate the effectiveness of the feature sets, we set up experiments on the problem of drug-usage Web page classification and university-related Web page classification. The impacts of the hierarchical headings on

different learning algorithms are shown in the next section.

6.1 University-related Data Sets

In the University-related data set, the Web pages have been hand-labeled into 4 categories, which are course homepage, faculty homepage, project homepage and student homepage. In this data set, some categories are actually closely related which make the classification more difficult. A course home page gives information about the subject such as the course outline, the class schedule, reference books. A faculty homepage is an instructor homepage, which gives information about instructor's research, teaching course. A project homepage is actually a research homepage. A student homepage is a personal homepage of a student in the university.

6.2 Drug-Usage data set

This data set is The Drug-Usaget data set consists of 353 Web pages corresponding to 5 categories in pharmaceutical domain. Those categories are about adverse, Clinical pharmacy, overdose, patient information and warning.

6.3 Experimental Setting

Each Web page is filtered to remove words that give no significance in predicting the class of the page. Then, the word stemming process is applied to each page by using Porter algorithm [5] in order to remove all suffixes and search for similar words based on the root word. Finally, we extract all headings appearing in each Web page to be the feature of the heading-based classifier. Therefore, each Web page can be viewed as a set of words appearing in the page's content and a set of words appearing in all headings.

The Results

Standard precision (P), recall (R), F_1-measure (F_1) are used to evaluate the performance of the classifiers[6]. These measurements are defined as follows.

$$P = \frac{\text{no. of correctly predicted examples in the target class}}{\text{no. of predicted examples in the target class}} \quad (7)$$

$$R = \frac{\text{no. of correctly predicted examples in the target class}}{\text{no. of all examples in the target class}} \quad (8)$$

$$F_1 = \frac{2PR}{P+R} \quad (9)$$

6.4 Experimental results of Supervised Naive Bayes Classifier.

The objective of this experiment is to see how the hierarchical headings structure effects the performance of supervised naive Bayes classifier (S-Bays). In the Table 4 and 5, feature A means that the words appearing in the headings are treated as a plain text (no weight assignment). Feature set B means that we make use of the hierarchical headings structure. (assign the weight according to the weight presented in Table 1). Feature set C has the same meaning as feature set B except that we discard the html tag that change the style of the words; eg. <i>, <I> , <b>.

Table 4. The F_1 performance on Drug Usage Data set

S-Bayes	F_1-Measure (%)				
feature	3726	6109	7918	14187	41607
A	34.63	59.93	80.91	85.30	94.02
B	37.42	69.64	78.64	83.45	88.94
C	35.64	66.80	78.65	80.38	88.07

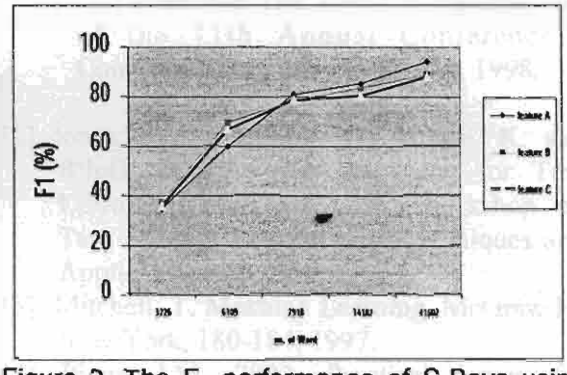
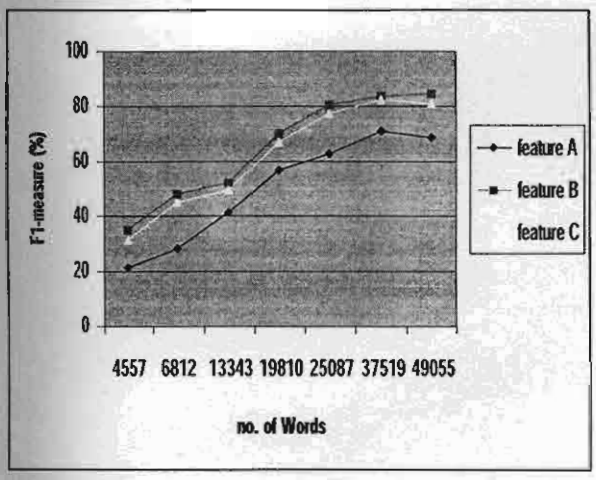


Figure 2. The F_1 performance of S-Bays using different no of words in the training data in the drug usage data set.

Table 5. The F_1 performance on University-related Data set

S-Bayes	F_1-Measure (%)						
feature	4557	6812	13343	19810	25087	37519	49055
A	21.34	28.34	41.36	56.75	62.85	70.82	68.81
B	34.88	48.25	52.46	70.18	80.60	83.42	84.56
C	31.42	44.79	49.76	66.72	77.14	82.27	81.10

Figure 3. The F_1 performance of S-Bays using different no of words in the training data on University-related data set.



From the experimental result, we found that the hierarchical headings feature set (feature B) could enhance the classifier 's performance in both data sets. As the number of words in the labeled training data increase, the performance of the naive Bayes classifier also increase.

6.5 Experimental results using hierarchical headings structure on different algorithms.

In this experiment, we investigate the impact of the hierarchical headings structure and see its contribution on different algorithms.

Table 6. The F_1 performance on Drug Usage Data set

Algorithm	F_1-Measure (%)				
	3726	6109	7918	14187	41607
S-Bays	37.42	69.64	78.64	83.45	88.94
I-ICT	60.21	78.28	84.00	92.59	90.12
Co-Training	70.76	79.21	76.07	82.32	81.92

Table 7. The F_1 performance on University-related Data set

	F_1-Measure (%)						
feature	4557	6812	13343	19810	25087	37519	49055
S-Bayes	34.88	48.25	52.46	70.18	80.60	83.42	84.56
I-ICT	44.98	49.25	55.46	78.18	81.60	84.42	85.56
Co-Training	41.42	64.79	69.76	76.72	79.14	82.27	84.10

7. Discussion

The experimental result (as shown in Table 6 and 7) shows that the I-ICT algorithm can gain the benefit of the hierarchical structure of the headings. I-ICT outperforms S-Bays in all experiment using different no of words in training data. This means that the classification mechanism of I-ICT has a high potential and has good strategy to view the Web page as the words appeared in the hierarchical heading and words appeared in the content. I-ICT got the higher correctness compare to Co-Training when using the no. of word more than 3726.

8. Conclusion

In this paper, we have proposed to use the hierarchical heading structure and demonstrated the concept of the I-ICT algorithm. Our algorithm has an advantage over the supervised learning algorithm in the sense that the classifier needs only small amount of initial labeled data, whereas the supervised learning algorithm needs a huge number of labeled data.

9. References

[1] Blum, A. and Mitchell, T. Combining labeled and unlabeled data with Co-Training, Proc. of the 11th Annual Conference on Computational Learning Theory, 1998.

[2] Jones, R., McCallum, A., Nigam, K. and Riloff, E. 1999. Bootstrapping for Text Learning Tasks. IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications. pp. 52-63.

[3] Mitchell, T. Machine Learning, McGraw-Hill. New York, 180-184, 1997.

[4] Pierre J.M. 2000. Practical Issues for Automated Categorization of Web Sites.

[5] Porter, M. F.. An algorithm for suffix stripping, 130-137, 1980

[6] van Rijsbergen, C.J., Information Retrieval, Butterworths, London, 1979.