อัลกอริทึมการสอนไขว้แบบวนซ้ำสำหรับการจำแนกประเภทเว็บเพจ

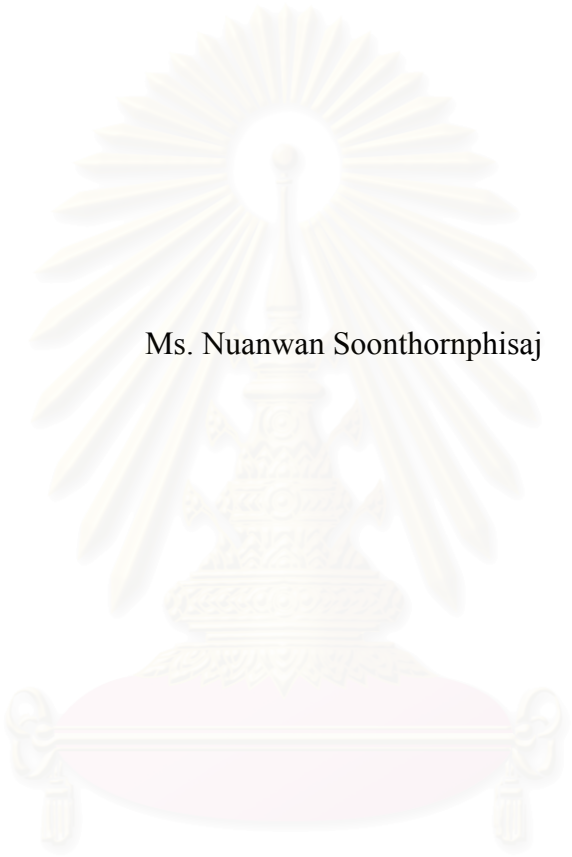
นางนวลวรรณ สุนทรภิษัช

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2545

ISBN 974-17-1361-4

AN ITERATIVE CROSS-TRAINING ALGORITHM FOR WEB PAGE CATEGORIZATION

Ms. Nuanwan Soonthornphisaj



A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic year 2002

ISBN 974-17-1361-4

Thesis Title	An Iterative Cross-Training Algorithm for Web Page Categorization
By	Ms.Nuanwan Soonthornphisaj
Field of Study	Computer Engineering
Thesis Advisor	Assistant Professor Boonserm Kijsirikul, D.Eng.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctor's Degree

…….Dean of Faculty of Engineering
(Professor Somsak Punyakeow, D.Eng.)

THESIS COMMITTEE

……. Chairman
(Associate Professor Prabhas Chongstitvatana, Ph.D.)

…… Thesis Advisor
(Assistant Professor Boonserm Kijsirikul, D.Eng.)

…… Member
(Associate Professor Wanchai Rivepiboon, Ph.D.)

……..Member
(Arthit Thongtak, D.Eng.)

……..Member
(Associate Professor Peter Haddawy, Ph.D.)

นวลวรรณ สุนทรภิษัช : อัลกอริทึมการสอนไขว้แบบวนซ้ำสำหรับการจำแนกประเภทเว็บเพจ. (AN ITERATIVE CROSS-TRAINING ALGORITHM FOR WEB PAGE CATEGORIZATION) อ. ที่ปรึกษา : ผศ.ดร. บุญเสริม กิจศิริกุล 68 หน้า. ISBN 974-17-1361-4.

เป้าหมายของการจำแนกประเภทเว็บเพจคือการระบุประเภทของเว็บเพจให้อยู่ในหมวดหมู่ที่กำหนด ซึ่งในงานวิจัยก่อนหน้านี้ได้มีการศึกษาถึงเทคนิคในการจำแนกประเภทเว็บเพจให้มีประสิทธิภาพมากที่สุด หนึ่งในวิธีการดังกล่าวคือการใช้อัลกอริทึมการเรียนรู้แบบ Supervised Learning ซึ่งต้องอาศัยข้อมูลฝึกเป็นจำนวนมาก ปัญหาที่เกิดขึ้นคือการสร้างชุดข้อมูลฝึกเหล่านี้ต้องอาศัยกำลังของผู้เชี่ยวชาญเฉพาะด้านต่างๆเป็นจำนวนมาก อีกทั้งยังต้องใช้เวลาเป็นจำนวนมากในการสร้างชุดข้อมูลดังกล่าว ดังนั้นจึงมีความจำเป็นที่จะต้องคิดค้นอัลกอริทึมที่สามารถแก้ปัญหาดังกล่าวได้โดยที่ยังคงประสิทธิภาพการทำงานให้ใกล้เคียงกัน ดังนั้นผู้วิจัยจึงได้เสนออัลกอริทึมการสอนไขว้แบบวนซ้ำสำหรับการจำแนกประเภทเว็บเพจขึ้นเพื่อแก้ปัญหาดังกล่าว

ผู้วิจัยได้ทำการทดสอบอัลกอริทึมการสอนไขว้แบบวนซ้ำเพื่อการจำแนกประเภทเว็บเพจกับข้อมูลทั้งสิ้น 4 ชุด ประสิทธิภาพการทำงานของอัลกอริทึมได้ถูกนำไปวิเคราะห์และเปรียบเทียบกับอัลกอริทึมอื่นๆ ได้แก่ Supervised Learning, Co-Training และ Expectation Maximization ผลการทดลองพบว่าโดยเฉลี่ยแล้วอัลกอริทึมการสอนไขว้แบบวนซ้ำเป็นอัลกอริทึมที่มีประสิทธิภาพทัดเทียมกับอัลกอริทึมแบบ Supervised Learning และมีประสิทธิภาพดีกว่าอัลกอริทึมแบบ Co-Training และ Expectation Maximization

ผู้วิจัยได้ศึกษาถึงผลกระทบของข้อมูลสัญญาณรบกวน (Noise) ที่มีต่อการจำแนกประเภทเว็บเพจ จากผลการทดลองพบว่าในกรณีที่มีการให้ความรู้ที่เกี่ยวกับลักษณะของปัญหา (domain knowledge) แก่อัลกอริทึม อัลกอริทึมการสอนไขว้แบบวนซ้ำไม่สูญเสียประสิทธิภาพการทำงานเมื่อเทียบกับอัลกอริทึมอื่นๆ แต่ในกรณีที่ไม่มีการให้ความรู้ที่เกี่ยวกับลักษณะของปัญหาแก่อัลกอริทึม พบว่าอัลกอริทึมการสอนไขว้แบบวนซ้ำสูญเสียประสิทธิภาพการทำงานน้อยกว่าวิธีการอื่น

นอกจากนี้ผู้วิจัยได้ปรับปรุงประสิทธิภาพของอัลกอริทึมโดยการนำวิธีการโปรแกรมตรรกะเชิงอุปนัย (Inductive Logic Programming) มาประยุกต์เพื่อให้อัลกอริทึมการสอนไขว้แบบวนซ้ำทำงานได้อย่างมีประสิทธิภาพมากยิ่งขึ้น ซึ่งผลการทดลองพบว่า วิธีการโปรแกรมตรรกะเชิงอุปนัยมีส่วนช่วยในการทำให้ประสิทธิภาพการทำงานโดยรวมของ อัลกอริทึมการสอนไขว้แบบวนซ้ำดียิ่งขึ้น

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา 2545

4271806521 : MAJOR COMPUTER ENGINEERING

KEY WORD: MACHINE LEARNING / WEB PAGE CATEGORIZATION

NUANWAN SOONTHORNPHISAJ: AN ITERATIVE CROSS-TRAINING ALGORITHM FOR WEB PAGE CATEGORIZATION. THESIS ADVISOR : ASST. PROF. BOONSERM KIJSIRIKUL, D.Eng., 68 pp. ISBN 974-17-1361-4.

The goal of the Web page categorization is to classify Web documents into a certain number of predefined categories. 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 labeled training documents. Though it is difficult to manually categorize unlabeled documents for creating training data, it is easy to collect unlabeled ones. Therefore, a new machine learning algorithm is investigated to overcome these difficulties and effectively utilize unlabeled documents. We propose in this thesis a novel approach called *Iterative Cross-Training* (*ICT*) to solve the Web page categorization problem.

In this thesis, we applied the algorithm to solve the Web page categorization problems on four data sets. The performance of ICT was evaluated and analyzed with the supervised learning, Co-Training and Expectation Maximization algorithms. We found that the ICT algorithm is an effective approach for the Web page categorization task.

We studied the effect of noise on the Web page categorization problem and found that the ICT algorithm was robust to noise when domain knowledge was given. In case that no domain knowledge was available, ICT's performance loss was less than other learning algorithms.

Furthermore, the enhanced version of ICT was developed. We integrated an Inductive Logic Programming (ILP) with the ICT algorithm. The experimental results showed that the ILP system had capability to increase the overall performance of ICT.

Department	Computer Engineering	Student's signature ……
Field of study	Computer Engineering	Advisor's signature ……
Academic year	2002	

ACKNOWLEDGEMENTS

I owe many thanks to a number of people who have supported, inspired, and motivated me throughout my graduate study. First, I would like to thank my thesis advisor, Assistant Professor Boonserm Kijsirikul, who provided me with a great deal of guidance in the area of machine learning. He always pushes and motivates me throughout this study period. His valuable suggestions and comments made this work feasible.

I would like to thank my thesis committee members, Associate Professor Prabhas Chongstitvatana, Associate Professor Wanchai Rivepiboon, Associate Professor Peter Haddawy and Dr. Arthit Tongtak, who provided valuable comments at the committee meetings

I would like to thank Dr. Colin Duerkop and Dr. Beatrice Gorawantshy from the Konrad Adenauer Foundation for the research scholarship in Germany. I would also like to extend my appreciation to Professor Luc De Raedt from the Machine Learning and Natural Language Laboratory, University of Freiburg who gave me many valuable suggestions on my publication.

I especially thank to my big supporter, Chaiyong Soonthonphisaj, who works like my personal secretary. All of this works would not have been possible without his encouragement and assistance. Special thank goes to my little boy, Chanon Soonthornphisaj for his patience during my study.

I also would like to thank a nice guy, Dr. Nuttakorn Thubthong, from the Department of Physics, Chulalongkorn University, who was a great source of helps and advices. My thank also goes to Dr. Sukree Sinthupinyo and Ratthachat Chatpatanasiri for their helps and discussions on my thesis. Most of all, I thank all members of the Machine Intelligence and Knowledge Discovery Laboratory, including all friends in the doctoral degree program, who made my time at Chulalongkorn University wonderful and memorable.

My special thanks go to my parents for their supports and encouragements throughout my graduate study. The success of my study was originated from their visions, which should never be forgotten.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

CHAPTER 1

INTRODUCTION

The availability of large, heterogeneous repositories of Web pages is increasing rapidly. There are billions Web pages accessible on the Internet with 1.5 million pages being added daily (Pierre 2000). A user searching for documents within a specific category using a general purpose search engine might have a difficult time finding valuable documents. Search engine Web sites, such as Yahoo[1] organize their Web resources in category-specific style. These Web sites currently use human experts to categorize the documents. However, the number of Web pages nowadays is exponentially increasing. 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.

In this chapter, we discuss what we mean by Web page categorization, motivation, objectives and contributions of the thesis.

1.1 Web Page Categorization

The purpose of Web page categorization is to assign each Web page to an appropriate category, based on the content of the document. The list of categories is predetermined. A stream of Web pages must be entered to the classifier, and then the classifier assigns categories that represent the content of the pages.

The Web page categorization task is typically difficult because the category assignment is based on the meaning of the content. Therefore, the system should be able to understand the document correctly.

In this thesis, we develop and investigate a new algorithm called *Iterative Cross-Training* (*ICT*) for automatic Web page categorization using machine learning approach. The reason that we choose a machine learning technique is that the system itself is capable to learn how to solve the problem. A learning algorithm needs a set of labeled documents (documents together with categories). Then the algorithm analyses these examples, extracts and generalizes the knowledge and keeps this knowledge as an output. This knowledge can be used to classify new unseen documents.

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

1.2 Motivation

Due to the huge amount of data rapidly accumulated in the World Wide Web, surfing the Web to get information becomes problematic. One solution to this problem is to organize all Web documents into categories. Search engine Web sites employ process known as robots, spiders or crawlers to recursively traverse the Web space and retrieve Web documents. To keep updating the index of Web documents and organize them into categories, the search engines should have an automatic Web page categorization system in order to cope with the problem.

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 easy to manually categorize them for creating training documents. Therefore, a new machine learning algorithm should be investigated to overcome these difficulties.

1.3 Objectives

This thesis aims to accomplish the following tasks.

- To develop an algorithm in order to deal with the Web page categorization problem.
- To empirically study the performance of the algorithm under different problem conditions.
- To find appropriate features of Web pages which have a high impact on the categorization process.

1.4 Scope and Limitation

Data sets concerning our experiments are Web pages (in html format). The feature sets extracted from the Web pages are text-based features. That means, the images, pluged-in applications and other non-text media are not taken into account. The outcome of the research will be a classifier with a learning ability to classify Web pages into predefined categories.

1.5 Contributions

The goal of this thesis is to develop a new learning algorithm, which has an ability to solve Web page categorization problems. The developed algorithm is empirically studied and analyzed under different problem conditions.

1.6 Organization of the Thesis

In this thesis, we address the problems and the challenges of the automatic Web page categorization using machine learning techniques. The remainder of the thesis is organized into eight chapters as follows.

Chapter 2 reviews several topics related to this work. We give the background of machine learning techniques followed by the description of various algorithms, which were applied by other researchers in Web page categorization problems.

Chapter 3 introduces our proposed algorithm called *Iterative Cross-Training* algorithm (*ICT*). We start by presenting the motivation and the problem specification of *ICT*. Next, we describe the detail of the ICT algorithm.

Chapter 4 investigates the behaviour of ICT concerning the availability of domain knowledge. We apply the ICT algorithm to solve the Thai/non-Thai Web page identification problem.

In Chapter 5, we apply the ICT algorithm to solve the Web page categorization problem on three data sets.

Chapter 6 presents the study of ICT using different numbers of initial labeled data. In the case that no domain knowledge is available, ICT needs a small amount of initial labeled data to start up the learning process. We vary different numbers of these labeled data to observe the performance of our algorithm. Furthermore, we evaluates the performance of ICT in the presence of noisy labeled data. In reality, this problem is difficult to avoid, and therefore it is interesting to see how ICT is affected by this problem.

Chapter 7 gives detail about the performance enhancement of ICT using *Inductive Logic Programming* (*ILP*). We encapsulate the background knowledge in the first-order representation and employ an ILP system to learn to construct the rules, which can differentiate the class of Web pages.

Chapter 8 summarizes our works and gives directions for future works.

CHAPTER 2

BACKGROUND

In this chapter, we review several topics which are related to our work. We start by providing the machine learning concept. We then review various traditional machine learning algorithms such as *k*-nearest neighbor, decision tree learning, rule set learning, support vector machines, etc. These algorithms have been applied to the text categorization problem. In the rest of this chapter, we will describe some of these algorithms and their contributions on the text categorization problem. In addition, the evaluation criteria of text classifiers are provided in the last section.

2.1 Machine Learning Techniques

Since the early ’90s, machine learning approach has gained popularity in the text categorization field. In this approach, a general inductive process (learner) automatically builds a classifier for a category by observing the characteristics of a set of documents manually labeled by the experts. This classifier is expected to have an ability to categorize a new unseen document correctly. There are three common types of learning: supervised, semi-supervised and unsupervised learning.

2.1.1 Supervised Learning

Learning is supervised when the learner is supplied with the preclassified documents called labeled data. As shown in Figure 2.1, after the preprocessing is completed, the learner finds the common properties of these labeled data and constructs a model (knowledge) for each class. Then the classifier employs the model to predict the class of test data. The test data is used to evaluate the performance of the classifier on unseen data. One of the problems of supervised learning is overfitting (Witten and Frank 2000). The classifier works well on training data but performs poorly on the test data.

The usual way to evaluate an ability of the learner is to do a *k*-fold cross validation approach (Mitchell 1997). In *k*-fold cross validation, the data is divided into *k* subsets of equal size. The model is trained *k* times, each time leaving out one of the subsets for testing.

Several supervised learning algorithms have been employed to solve the text categorization problem, i.e. *k*-nearest neighbor, decision tree learning, naive Bayes algorithm, etc.

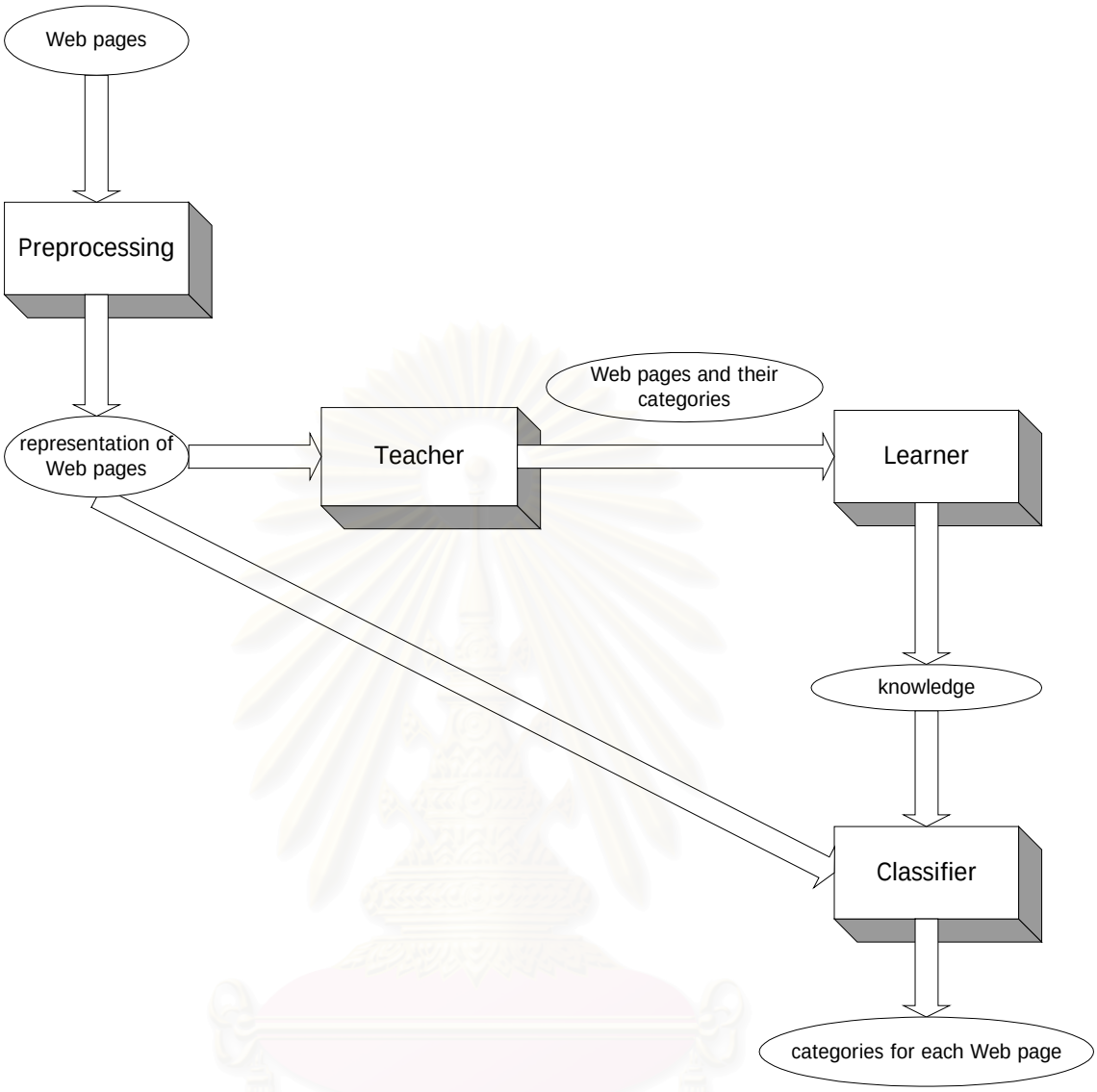


Figure 2.1: Supervised learning model.

2.1.1.1 *K*-Nearest Neighbor

K-nearest neighbor (*k*-NN) classification is an instance-based learning algorithm that has been applied to text categorization since the early days of the research. In this classification paradigm, *k*-nearest neighbors of a test document are computed to predict the class of the document. Then the similarities, measured based on *cosine* or *Euclidean distance* functions from this document to the *k*-nearest neighbors, are summarized according to the class of the neighbors. Then the test document is assigned to the most similar class. A major drawback of the similarity measurement used in *k*-NN is that it uses all features equally in computing similarities. This method leads to the poor similarity measures and classification errors, when only a small subset of the words is useful for classification. To cope with this

problem, a variety of techniques have been developed for adjusting the importance of the terms in the document, e.g. weighting based on mutual information (Aha 1997). Another problem is the difficulty in deciding an optimal *k* value. Normally, the researchers will conduct a series of experiments using different value of *k*.

Han et al., proposed a weight adjusted *k*-nearest neighbor (WAKNN) classification algorithm to deal with the text categorization problem (Han et al. 2001). Each document is represented by the vector of words. Initially, each word is assigned a weight according to its mutual information value. Then the weighted cosine similarity is calculated between each training document and the test document. The next step is to calculate the goodness of this initial weight vector using an objective function and update the weight value that gives the best performance on the objective function. They conducted experiments on four data sets. The experimental results showed that the algorithm outperformed the traditional *k*-NN.

Yang investigated the performance of variety classification algorithms on Reuter data set[1] and found that *k*-NN (k = 30, 45, 65) exhibited the best performance for the text categorization task. *K*-NN achieved 82% of correctness and was considered to be a challenging method (Yang 1998).

2.1.1.2 Decision Tree Learning

Decision tree learning is a well-known approach for the classification problems. The algorithm classifies an instance by sorting them down the tree from the root to the leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance and each branch descending from that node corresponds to one of the possible values for this attribute. A new instance is classified by starting at the root node of the tree, testing the attribute specified by the node, then moving down the tree branch corresponding to the value of the attribute in the given example (Mitchell 1997). The success of classification using decision trees depends heavily on the attributes or features, which represent the concept of category. Therefore, the feature selection process is the most important part, when we use a decision tree as a classification algorithm. In the text categorization framework, the first step is to produce a list of attributes from the labeled data set. These attributes are single words or word phrases. Given an attribute list, labeled documents can be described in terms of the words or phrases found in the documents. Each document consists of the values of the attributes, where the values are boolean indicating whether the attribute appears in the text or not. After the tree construction, the rule induction

[1] Reuter collection http://moscow.mt.cs.cmu.edu:8081/reuters_21450/parc/

process is started. The objective is to find sets of decision rules that have the ability to distinguish one category from the others.

Apte et al. applied decision trees to the text categorization. The feature selection is based on the number of occurrences appeared in the corpus (Apte et al. 1998). They extracted 10,000 most frequent words and pruned down the attribute size using the entropy-based technique. The result shows that the decision tree gives the accuracy at 67% on Reuter data set. They proposed to use the rule induction technique called Swap-1 (Weiss and Indurkhya 1993) to extract the rules from the tree and could get the accuracy at 80.5%.

One problem of the decision tree is the overfitting, where the tree is too specialized to the training data. Researchers have observed that the variance of training data can be reduced by constructing many decision trees using sampling technique. Breiman proposed to use the technique called bagging that randomly selects sample sets of size *n* from the training data to construct the trees (Breiman 1996). Chen constructed a collection of decision trees called decision forest (Chen and Kam 2000). The feature selection is based on Document Frequency threshold. The feature vector was created for each document. Each element in the vector is the product of *term frequency* and *inverse document frequency* of each term. These trees are constructed by randomly selecting subsets of the components of the feature vector. It has been demonstrated that this method outperforms single decision tree using all of the available features.

2.1.1.3 Naive Bayes Algorithm

Naive Bayes is a simple but effective text classification algorithm for learning from labeled data (Lewis 1999). The algorithm finds the model of the class from the labeled data. The classification's criteria is based on two parameters, which are the class prior probabilities and the posteriori probabilities that are estimated for the class-conditional probabilities for each word in the vocabulary, V, from labeled training data D. This is done by counting the frequency that word w_t occurs in all word occurrences for documents d_i in class c_j, supplemented with Laplace smoothing to avoid zero probability.

$$P(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} N(w_t, d_i) P(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) P(c_j \mid d_i)} \tag{2.1}$$

where $N(w_t,d_i)$ is the count of the number of times word w_t occurs in document d_i.

$P(c_j|d_i) \in \{0,1\}$ as given by the class label.

The prior probability of each class is calculated in a similar fashion, counting over documents instead of words:

$$P(c_j) = \frac{1 + \sum_{i=1}^{|D|} P(c_j \mid d_i)}{|C| + |D|} \tag{2.2}$$

During the classification time, we use these estimated parameters by applying Bayes' rule to calculate the posterior probability of each class label and taking the most probable class as the prediction. This makes use of the naive Bayes independence assumption, which states that words occur independently of each other, given the class of the document:

$$P(c_j \mid d_i) \propto P(c_j) P(d_i \mid c_j)$$

$$= P(c_j) \prod_{k=1}^{|d_i|} P(w_{d_i,k} \mid c_j) \tag{2.3}$$

The word independence assumption causes naive Bayes to predict extreme (nearly 0 or 1) posterior class probabilities. However, while these estimates are poor, naive Bayes classification accuracy is typically high. This can be explained because classification is only a function of which class has the maximum posterior, and is not concerned with its actual value (Domingo and Pazzani 1997).

McCallum and Nigam investigated the probabilistic framework of naive Bayes and applied the algorithm on Web page classification problem (McCallum and Nigam 1998). They found that, the algorithm could achieve the correctness at 74%.

2.1.1.4 Rule Learning Algorithm

The famous rule learning algorithm called *Ripper* has been extensively applied to the text categorization problem. The classifier constructed by *Ripper* is a set of rules, which is learned for each category. This set of rules can be interpreted as a disjunction of conjunctions: for instance, a document *d* is considered to be in the category *music* if and only if

(the word `song` appears in *d*) OR
(the word `singer` appears in *d* AND the word `rhythm` appears in *d*) OR
…
(the word `song` appears in *d* AND the word `rhythm` appears in *d*)

Rule sets enjoy some properties that make them useful in certain contexts. If a rule set is compact, it is relatively easy for people to understand; this may make it easier for users to accept a learned classifier as being reasonable. Rule sets can also be easily converted to

queries for a boolean search engine (Cohen and Singer 1998). There are a number of subtleties involved in learning rule sets. Due to the greedy algorithm style in *Ripper*, the algorithm tends to give rule sets that have high error rates. Furthermore, many algorithms that produce rule sets tend to be relatively inefficient for noisy data sets. Because of these problems, the algorithm used in Ripper is relatively complex.

Ripper consists of two main steps. The first step is a greedy process, which is a set covering algorithm that constructs an initial rule set. It constructs one rule at a time and removes all examples covered by a new rule as soon as the rule is constructed. The heuristic used in construction of a rule is intended to ensure that the rule covers many positive examples and few negative examples. A rule is extended by repeatedly adding conditions to the rule to produce a longer and more specialized rule. The condition added is the one that yields the largest *information gain* (Quinlan 1990).

After growing the rule, the rule is pruned. The algorithm considers deleting any final sequence of conditions from the rule and chooses the deletion that maximizes the function

$$f(r_i) = \frac{U^{+}_{i+1} - U^{-}_{i-1}}{U^{+}_{i+1} + U^{-}_{i+1}} \tag{2.4}$$

where U^{+}_{i+1} and U^{-}_{i+1} are the numbers of positive examples and negative examples, respectively, in the pruning set covered by the new rule. After pruning, the pruned rule is added to the rule set and the examples covered by it are removed.

The second step in *Ripper* is the rule set optimization process. After *Ripper* stops adding rules, the rule set is optimized so as to further reduce its size and improve its accuracy. Rules are considered in turn in the order in which they were added. For each rule *r*, two alternative rules are constructed, the *replacement* rule and the *revision* rule. The replacement for *r* is formed by growing and then pruning a rule r', where pruning is guided so as to minimize error of the entire rule set (with r' replacing *r*) on the pruning data. The revision of *r* is formed the same way as the replacement rule, except that it is grown by greedily adding literals to *r*. Finally a decision is made as to whether the final theory should include the revised rule, the replacement rule, or the original rule. After the optimization process has passed, the rule set may cover fewer positive examples; thus the first step is called again on the uncovered positive examples.

Cohen et al. performed experiments on two text categorization data sets: a corpus of AP titles and a corpus of news stories. They found that the rule-learning algorithm outperforms the traditional method (Cohen et al. 1996).

2.1.1.5 Rocchio's Algorithm

This algorithm was first introduced by Rocchio in 1971 (Rocchio 1971). It uses the *tfidf* weight for each word. The *tfidf* is one of the most successful weighting schemes in Information Retrieval (Pazzani 1997). The computation of the weights reflects empirical observations regarding text. Terms that appear frequently in one document (*tf* = term-frequency), but rarely occur in other documents (*idf* = inverse-document-frequency), are more likely to be relevant to the topic of the document. Therefore, the *tfidf* weight of a term in one document is the product of its term-frequency (*tf*) and the inverse of its document frequency (*idf*). In addition, to prevent longer documents from having a better chance of retrieval, the weighted term vectors are normalized to unit length.

Many researchers have applied Rocchio's algorithm to text categorization problem. Pazzani et al.'s work compared several classification algorithms on Web page identification problem and found that Rocchio's algorithm performed well on most domains (Pazzani 1997).

2.1.1.6 Support Vector Machines

Support Vector Machines (SVMs) are based on the Structural Risk Minimization principle (Vapnik 1995). It was first introduced by Vapnik in 1995. The idea of structural risk minimization is to find a hypothesis *h* for which we can guarantee the lowest true error. The true error of *h* is the probability that *h* will make an error on an unseen and randomly selected test example. An upper bound can be used to connect the true error of a hypothesis *h* with the error of *h* on the training set and the complexity of *H* (measured by VC-Dimension), the hypothesis space containing *h* (Vapnik 1995). One remarkable property of SVMs is that their ability to learn can be independent of the dimensionality of the feature space. SVMs measure the complexity of hypotheses based on the margin with which they separate the data, not the number of features. This means that the algorithm can generalize even in the presence of many features.

An SVM method has been introduced in text categorization by Joachims (Joachims 1998). He used polynomial and RBF kernels to compare the performance with four conventional learning methods commonly used for text categorization. These algorithms are the naive Bayes classifier, the Rocchio algorithm, *k*-NN and C4.5 decision tree learner. The experiments were conducted by using the software package called SVM light[2] and tested on the Reuter data set and the Ohsumed collection. He found that SVMs outperformed all

[2] SVM light http://www-ai.informatik.uni-dortmund.de/thorsten/svm_light.html

conventional methods significantly. Note that, the training time of SVMs is more expensive than the conventional methods. SVMs offer two important advantages for the text categorization problem:

- term selection is often not needed, as SVMs tend to be fairly robust to overfitting and can scale up to considerable dimensionalities;
- no human and machine effort in parameter tuning on a validation set is needed.

SVMs were subsequently used in (Drucker et al. 1999; Dumais et al. 1998; Dumais and Chen 2000; Klinkenberg and Joachims 2000; Taira and Haruno 1999; Yang and Liu 1999).

2.1.2 Semi-Supervised Learning

In many learning tasks, there is a large supply of unlabeled data but insufficient labeled data since it can be expensive to generate. Semi-supervised learning combines labeled and unlabeled data during the training process to improve performance. In supervised classification, there is a known, fixed set of categories and category-labeled training data is used to induce a classification function. In semi-supervised classification, training also exploits additional unlabeled data, frequently resulting in a more accurate classification function.

Semi-supervised or boosting algorithm was first introduced by Freund and Schapire (Freund and Schapire 1996). The algorithm requires only a small amount of labeled data as a seed information. It is believed that unlabeled data can be used in some settings to improve classification, although it is exponentially less valuable than labeled data. Anyway, we can not learn to classify unseen data based on only unlabeled data. Therefore, unlabeled data must be coupled with at least some information about the target function for the learning task.

The algorithm called bootstrapping was investigated in the domain of text learning by Jones et al. (Jones et al. 1999). This algorithm needs knowledge about the classes of documents, which is provided in the form of a few keywords per class and a class hierarchy.

2.1.2.1 Co-Training Algorithm

The Co-Training algorithm was first introduced by Blum and Mitchell in 1998 (Blum and Mitchell 1998). The concept of the algorithm is based on the boosting technique. That means, the algorithm learns from a small number of initial labeled data, and 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 text appearing on the hyperlink or the text appearing in the page content.

Table 2.1 : 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 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'*.

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. The Co-Training algorithm is shown in Table 2.1.

From the table, there are two classifiers, *Classifier1* and *Classifier2*, each of which learns from a set of labeled training examples, *LE*. The learning process starts by randomly selecting *u* examples from a set of unlabeled examples and adding these examples into a pool *UE'*. These *u* examples will be labeled by *Classifier1* and *Classifier2*. These two classifiers select *p* positive and *n* negative examples to be added into the labeled data, *LE*. Then the algorithm refills a set of *UE'* by randomly choosing *2p+2n* examples from *UE*. The learning process is continued until all of unlabeled examples are labeled.

2.1.2.2 Expectation-Maximization

Another boosting style algorithm is *Expectation-Maximization* (*EM*). This algorithm was first introduced by Dempster et al. (Dempster et al. 1977). It is an iterative algorithm for maximum likelihood estimation in problems with incomplete data. Given a model of data

Table 2.2: The training algorithm for the naive Bayes classifier using the EM algorithm.

Given:

- a set *UE* of unlabeled examples

Initialize the parameter set of *Classifier*1 to θ_{l0} to label *UE*.

$$\theta_l \rightarrow \theta_{l0}$$

Use the labeled examples in *UE* to estimate the parameter $Pr(c_i|l_j)$ and $Pr(l_j)$ of the *Classifier*2 with $Pr(l_j|d) \in \{0,1\}$.

Loop until the parameters of *Classifier*2 do not change or the number of iterations exceeds a predefined value:

- (E-step) Estimate the probabilistically-weighted class labels, $Pr(l_j|d)$, for every document using Equation 2.7.
- (M-step) Use the estimated class labels, $Pr(l_j|d)$, to calculate new parameters using all documents, by Equation 2.5 and 2.6.

---

generation, and data with some missing values, EM iteratively uses the current model to estimate the missing values, and then uses the missing value estimates to improve the model. Using all of the available data, EM will locally maximize the likelihood of the parameters and give estimates for the missing values. Therefore, the class labels of the unlabeled data are treated as the missing values. EM has two steps, which are the E-step and M-step, respectively. The E-step calculates probabilistically weighted class labels for every document using the classifier. For the M-step, it estimates new classifier parameters using all documents. In Nigam, et al.'s work (Nigam et al. 1999), they combined EM with a naive Bayes classifier to solve the text classification problem. The algorithm has shown to be able to significantly increase text classification accuracy when given limited amounts of labeled data and large amounts of unlabeled data.

As shown in Table 2.2, the first step starts with the parameter estimation of the naive Bayes classifier which learns from initial labeled data, and then the classifier assigns the weighted class label to all unlabeled data. The training process is iterated with the E-step and M-step until the algorithm is converged.

To represent the content of a Web page, we use different representations to solve the problem of Thai Web page identification and Web page categorization. We employ *bag-of-characters* representation in the problem of Thai Web page identification in Chapter 4,

whereas for Web page categorization in Chapter 5 we use *bag-of-words* representation. We first describe EM for Thai Web page identification as follows.

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

$$Pr(c_i | l_j) = \frac{1 + \sum_{d \in D} N(c_i, d) Pr(l_j | d)}{T + \sum_{k=1}^{T} \sum_{d \in D} N(c_k, d) Pr(l_j | d)} \tag{2.5}$$

Where T is the total number of unique characters in the training set, $N(c_i,d)$ is the number of times character c_i occurs in document. The probability of a class label is given by Equation 2.6.

$$Pr(l_j) = \frac{1 + \sum_{d \in D} Pr(l_j | d)}{|L| + |D|} \tag{2.6}$$

Where $|L|$, $|D|$ are the number of class labels and the number of documents in the training set. Given an unlabeled document d, of n character $(c_1 , c_2 , \dots, c_n)$, the naive Bayes classifier estimates the probability that the document belongs to class label l_j by using Equation 2.7 below.

$$Pr(l_j | d) = \frac{Pr(l_j) Pr(d | l_j)}{Pr(d)} = \frac{Pr(l_j) \prod_{i=1}^{n} Pr(c_i | l_j)}{\sum_{k=1}^{|L|} Pr(l_k) \prod_{i=1}^{n} Pr(c_i | l_k)} \tag{2.7}$$

Note that now $Pr(l_j|d)$ is a probabilistically-weighted value; each document d is considered to be of class label l_j with probability equal to the estimate $Pr(l_j|d)$.

This kind of algorithm was also applied by Brin (Brin 1998) in order to extract book titles and author names from the World Wide Web. From a small set of seeded books, his DIPRE algorithm searches the Web for known pairs and learns new patterns that are common for these pairs. Then, these new patterns are used to identify new books.

McCallum et al. (McCallum et al. 1998) proposed to use a method called Query-by-Committee (QBC) in combination with EM to solve the text classification problem. The concept of active learning aims to select the most informative examples. The committee is the set of classifiers, which are created by sampling according to the distribution of classifier parameters specified by the training data. The experimental results show that the QBC with EM outperforms the baseline algorithm (EM) with 64% of correctness.

2.1.2 Unsupervised Learning

In unsupervised learning, an unlabeled data set is partitioned into groups of similar examples, typically by optimizing an objective function that characterizes good partitions. In this paradigm, the system does not need a predefined category for documents. The algorithm tries to produce clusters of documents, which have common characteristics. Therefore, this learning method is known as a clustering algorithm. There are several clustering algorithms applied in *Information Retrieval* (*IR*) field. Agglomerative hierarchical clustering and *k*-Means are two clustering techniques that are commonly used for document clustering. A widely known study, discussed in (Dubs and Jain 1998), indicated that agglomerative hierarchical clustering is superior to *k*-Means.

Document clustering has been investigated as a means of improving the performance of search engines by pre-clustering the entire corpus (van Rijsbergen 1979). A novel approach called *Suffix Tree Clustering* (*STC*) has been proposed in (Zamir and Etzioni 1998). They applied *STC* to a Web document clustering problem using only snippets and found that *STC* is faster and more effective than a standard clustering method.

2.2 Evaluation of Text Classifiers

The experimental evaluation of a classifier usually measures its effectiveness, which is an ability to take the right classification decisions. Classification effectiveness is normally measured in terms of the classic IR known as *precision*, *recall* and F_1.

2.2.1 Precision (*P*)

Precision is defined as the number of relevant documents retrieved divided by the total number of documents retrieved. For the classification problem, the classifier's task is to assign or predict the class for each example. Considering a given class as a positive class, the precision can be defined as in Equation 2.8.

$$\text{Precision} = \frac{\text{no. of correctly predicted positive examples}}{\text{no. of predicted positive examples}} \quad (2.8)$$

2.2.2 Recall (*R*)

Another standard measure in the IR field, recall, is defined as the number of relevant documents retrieved divided by the total number of relevant documents in the collection. Therefore, the recall can be defined as in Equation 2.9.

$$\text{Recall} = \frac{\text{no. of correctly predicted positive examples}}{\text{no. of all positive examples}} \quad (2.9)$$

2.2.3 F_1-measure

The F_1-measure was introduced by (van Rijsbergen 1979). It combines precision and recall with equal importance into a single parameter for optimization and is defined as in Equation 2.10.

$$F1 = \frac{2PR}{P+R} \quad (2.10)$$

In this chapter, related works in the text categorization problem were presented. We found that the text categorization is now a major research area. One of the reasons why the effectiveness of a text classifier has been improved, is the arrival of machine learning paradigm that provides more challenges to researchers in this field.

CHAPTER 3

ITERATIVE CROSS-TRAINING ALGORITHM

This chapter concentrates in details about our proposed method, which is *Iterative Cross-Training* (ICT). We first start with the motivation of ICT followed by the problem specification. Then we go through the description of the algorithm.

3.1 Motivation of ICT

The motivation of ICT algorithm is to utilize unlabeled examples during the learning process in order to enhance the classifier's performance. Given a set of labeled examples *LE*, a set of unlabeled examples *UE* and a hypothesis language *L*, the learning algorithm tries to find a hypothesis *H* within *L* that correctly classifies the examples in $LE \cup UE$. The expressiveness of the language and the used attributes will strongly influence the efficiency and effectiveness of the algorithm. This is the problem of learning from labeled and unlabeled data.

Consider the task of Thai Web page identification. We could either perform an expensive word segmentation algorithm before classifying the documents or alternatively perform a simple naive Bayes approach in order to classify the documents. The first approach allows identification of the words in the documents using the domain knowledge in the form of dictionary. The second employs bag-of-characters representation. Therefore hypotheses working within the first representation language promise to be more efficient than those working with the second one. On the other hand, those working within the second representation language are likely to be less accurate because they have no domain knowledge. Therefore we propose in this thesis a novel approach that combines these two representations.

3.2 Problem Specification

The idea of ICT is to learn simultaneously from two sets of hypotheses, one in each language. The problem specification of ICT can be formalized as follows:

Given:

- a set of labeled examples *LE*,
- a set of unlabeled examples, *UE*,
- two description languages of the examples L_1 and L_2,
- two learning algorithms A_1 and A_2; algorithm A_i works within hypothesis language L_i,

Find: hypotheses H_1 and H_2 (within L_1 and L_2) that correctly label the examples.

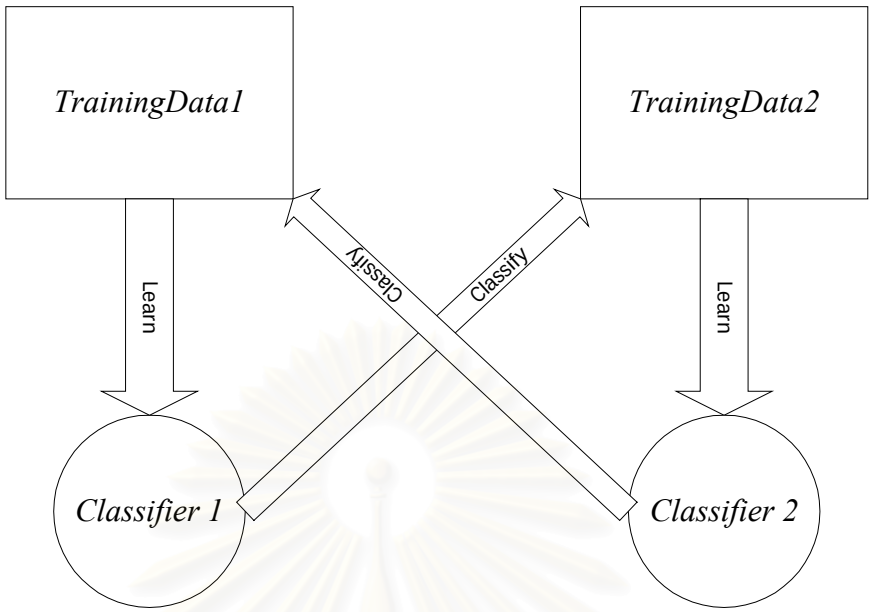


Figure 3.1: The Iterative Cross-Training algorithm

The idea is that the examples will be described in both description languages and that each of the algorithms (A_1 and A_2) will induce a hypothesis using the labeled examples only. The induced hypothesis is then employed to predict some of the labels of the unlabeled examples provided to the other learning algorithm. These examples are then used as an extended training set by the other learning algorithm. The induced hypothesis (H_1) predicts the labels of unlabeled examples and supplies these new labeled examples to A_2. The algorithm A_2 uses the extended training set to induce a new hypothesis and predicts the labels of the unlabeled examples and supplies these to the algorithm A_1. This learning process is repeatedly done in crossing style.

Figure 3.1 illustrates our learning algorithm, *Iterative Cross-Training* (*ICT*). The learning process of ICT starts with the parameter estimation of both classifiers. 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 examples are provided to the system, the values of the parameters can be predefined or randomly chosen ones.

Table 3.1: 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_{10}

$$\theta_1 \leftarrow \theta_{10}$$

Initialize the parameter set of *Classifier2* to θ_{20}

$$\theta_2 \leftarrow \theta_{20}$$

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

- Use *Classifier1* with the current parameter set θ_1 to label all data in *TrainingData2* as either positive examples or negative examples.
  - If no domain knowledge is given,

    check consistency of the classification with *Classifier2*.
- Train *Classifier2* by using labeled examples in *TrainingData2* to estimate the parameter set θ_2 of *Classifier2*.
- Use *Classifier2* with the current parameter set θ_2 to label all data in *TrainingData1* as either positive examples or negative examples.
  - If no domain knowledge is given,

    check consistency of the classification with *Classifier1*.
- Train *Classifier1* with the labeled examples in *TrainingData1* to estimate the parameter set θ_1 of *Classifier1*.

When an active 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. Otherwise, the active classifier will consider which classifier has more confidence in labeling this data item. If the active classifier has more confidence, it will label the example. In the case that the other classifier has more confidence than the active classifier, the example will not be labeled. Note that the confidence values are calculated based on the naive Bayes concept (see Equation 4.8). The purpose of the consistency checking is for producing more reliable labeled data, but the checking will slow down the learning process.

3.3 Sub-Classifiers in ICT

As stated in the previous section, ICT consists of two learners, which works together during the training process. These learners can be categorized into two classes as follows.

3.3.1 The Strong Learner

In this thesis, an algorithm supplied with background knowledge is denoted as a strong learner. This kind of learner is considered to have high performance due to the knowledge supplement. The learner usually consumes more computational time than the weak learner. The strong learners used in this thesis are as follows.

3.3.1.1 Word Segmentation Algorithm

The objective of a word segmentation algorithm is to separate a document's content into words. The algorithm is required when we deal with a language that has no word boundary delimiters, such as Thai language. In our experiment, we employ a word segmentation program developed by Meknavin et al. (1997). The strategy of this algorithm is to make use of a Thai dictionary to find the best segmentation. The dictionary is considered as background knowledge of the algorithm that has high potential in doing word segmentation. However, the algorithm takes a great deal of computational time during its process. We apply the word segmentation algorithm to the problem of Thai/non-Thai Web page identification. The mechanism of this algorithm is described completely in the next chapter.

3.3.1.2 Inductive Learning Algorithm

Among machine learning techniques, an *Inductive Logic Programming* framework (*ILP*) is one of the most effective approaches. In the ILP framework, the main idea is to obtain a set of generalized clauses that is general enough to cover the majority of the positive examples and sufficiently specific to rightly correspond to the concept we want to learn and to cover no (or a few – some noise can be allowed) negative examples (Sebillot et al. 2000). ILP is applied to many problem domains such as biology, biochemistry, Web mining, etc. (Finn et al. 1998; King and Sternberg 1990; Srinivasan and King 1999; Srinivasan et al. 1996; Craven et al. 1998). There are various ILP systems available such as Golem, Foil, Progol, etc. The detail of these systems will be given in Chapter 7. We apply a system called *Progol*[1], as a strong learner, to the Web page categorization problem.

[1] http://www.doc.ic.ac.uk/~shm/Software/

3.3.2 The Weak Learner

In this thesis, a weak learner means a learning algorithm, without a supplement of background knowledge, on the other hand supplied with only a small initial amount of labeled training data. The weak learner usually has less potential than a strong learner because it has no background knowledge and consumes fewer labeled examples compared to a supervised learning algorithm. We employ a naive Bayes algorithm to be the weak learner of our system. The naive Bayes has an advantage that its training process spends much less computation time. The detail of naive Bayes algorithm will be clearly explained in the next chapter.

CHAPTER 4

THAI WEB PAGE IDENTIFICATION

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 chapter, we first give the details of a word segmentation classifier followed by the naive Bayes classifier. We consider the word segmentation algorithm as a strong learner whereas the naive Bayes is a weak learner. Due to the fact that the word segmentation precess takes much more computational time than a simple naive Bayes approach, it is inefficient to use the word segmentation classifier in the real world application. Therefore, we employ ICT in order to have the word segmentation classifier boost the performance of the naive Bayes classifier.

4.1 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 expect that all words in the page appear in dictionary as the Web page usually contains names of persons, organizations, etc. not occurring in the dictionary and may contain words written in foreign languages. Therefore, it is necessary to determine how many words should be contained. This task is more difficult when dealing with a language that has no word boundary delimiters, such as Thai and Japanese (Meknavin et al. 1997).

Note that a string of Thai characters can usually be segmented in many possible ways as shown in Figure 4.1. Due to the fact that a word might be a substring of a longer word, and there is no a word delimiter in Thai, it is difficult to find which segmentation is correct. Below we describe the method for word segmentation. Given a Thai dictionary and a document d of n characters $(c_1,c_2,...,c_n)$, the word segmentation classifier generates all possible segmentations and finds the best segmentation $(w_1,w_2,...,w_m)$ that minimizes the cost function in Equation 4.1.

$$\underset{w_1,\ldots,w_m}{\operatorname{argmin}} \sum_{i=1}^{m} \text{cost}(w_i) \tag{4.1}$$

where $\text{cost}(w_i) = \eta 1$ 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, $c_i,...,c_j$, 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:

$$WordRatio = \frac{\text{the number of characters in all words}}{\text{the number of all characters in the document}} \quad (4.2)$$

The process of ICT applied to Thai Web page identification can be clearly explained as follows. 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 the *WordRatio* of a page is greater than the threshold, we will classify it as positive (a Thai page). Otherwise, we will classify it as negative (a non-Thai page). For simplicity, let us use only the threshold of *WordRatio* as the parameter of the word segmentation classifier.

Having only the threshold of *WordRatio* as the parameter of *classifier1* in Table 3.1, we can find a new *WordRatio* that produces more true positive and true negative examples for *TrainingData2*. As described above, most Thai Web pages should have a high value of *WordRatio*, whereas non-Thai Web pages should have a low value. If the number of Thai and non-Thai pages in *TrainingData2* are the same, it is easy to see that any value of *WordRatio* will give more correctly classified pages than incorrectly ones (except for *WordRatio* = 0.0 or *WordRatio* = 1.0, that gives the same number of correctly and incorrectly classified pages). In the case that the number of Thai pages is lower than the number of non-Thai pages, a high value of *WordRatio*, (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_{10} is for the case that the number of Thai pages is larger than that of non-Thai pages.

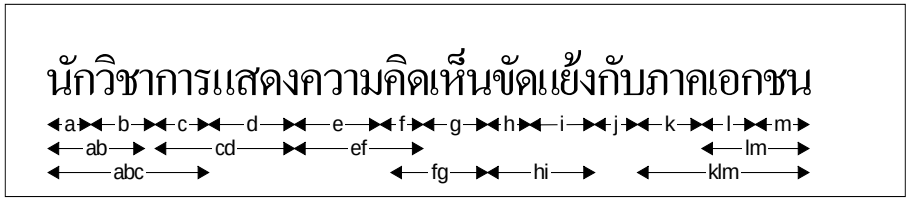


Figure 4.1: All possible word segmentations of a Thai sentence.

A new *WordRatio* can be estimated, after the naive 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 the case of $SP \geq LN$, the new *WordRatio* (θ_l) is estimated as:

$$\theta_l = \frac{SP+LN}{2} \tag{4.3}$$

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_l is estimated as:

$$\theta_l = \frac{V_{i*}+V_{i*+1}}{2} \tag{4.4}$$

$$V_{i*} = \underset{V_i}{\text{argmin}} \text{ (no. of } V_j + \text{ no. of } V_k) \tag{4.5}$$

Where V_k is a value of a labeled positive example, V_j is a value of a 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_l will completely separate the labeled positive from negative examples. Otherwise, θ_l will give the minimum errors of misclassified examples.

4.2 Naive Bayes Classifier (*Classifier2*)

For text classification, naive Bayes is among the most commonly used and the most effective methods (Mitchell 1997). 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 is very fast. In spite of its simplicity, the results below show the effectiveness of bag-of-characters representation in identifying Thai Web pages.

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

$$l^* = \underset{l_j}{\text{argmax}} \, Pr(l_j \mid c_1, \ldots, c_n) \tag{4.6}$$

$$= \underset{l_j}{\text{argmax}} \, \frac{Pr(l_j)Pr(c_1, \ldots, c_n \mid l_j)}{Pr(c_1, \ldots, c_n)} \tag{4.7}$$

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

$$l^* = \underset{l_j}{\text{argmax}}\ Pr(l_j) \prod_{i=1}^{n} Pr(c_i \mid l_j, c_1, \ldots, c_{i-1})$$
$$= \underset{l_j}{\text{argmax}}\ Pr(l_j) \prod_{i=1}^{n} Pr(c_i \mid l_j) \quad (4.8)$$

In this case, L is the set of positive and negative class labels. The term $Pr(c_1,\ldots,c_n)$ in Equation 4.7 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 naive Bayes are made (Mitchell 1997). 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. the significance of 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 naive Bayes has successfully been applied in various text classification problems (McCallum and Nigam 1998; Mitchell 1997; Yang and Pederson 1997). Using the above assumptions, Equation 4.7 can be simplified to Equation 4.8.

This model is also called the *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 the naive 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:

$$Pr(c_i|l_j) = \frac{1+ N(c_i,l_j)}{T + N(l_j)} \quad (4.9)$$

where $N(c_i,l_j)$ is the number of times character c_i appears in the training set from class label l_j, $N(l_j)$ is the total number of characters in the training set for class label l_j, and T is the total number of unique characters in the training set. Equation 4.9 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.

4.3 The Results on the Thai Web Page Identification

The objective of this experiment is to assess the effectiveness of ICT in the case that we supply the knowledge (in the form of dictionary) to a classifier (word segmentation classifier). We evaluate the performance of ICT in comparison with Co-Training, EM and supervised learning algorithms, respectively.

In the following subsections, we describe the data set and experimental setting for algorithms and the results.

4.3.1 Data Set & Experimental Setting

We collected the data set starting with 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 retrieved 450 pages. Therefore, we had approximately 900 Thai pages as Thai pages might link to ones which were in English or other languages. We also had 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 contained 600 pages (about 300 Thai, 150 Japanese and 150 English pages). Note that HTML mark-up tags were removed before the 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 without the consistency checking process. No labeled data was given to ICT. The initial θ_{10} was set to 0.7.

(2) For Co-Training, the values of the parameters of the classifier (in Table 2.1) were set in a similar way as in (Blum and Mitchell 1998). As Co-Training 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.

(3) For EM, we supplied the algorithm with 18 initial labeled data and 1182 unlabeled data.

(4) For the supervised naive Bayes classifier, we gave the algorithm 1200 initial labeled data.

4.3.2 Experimental Results

After the training process of ICT was finished, we evaluated the performance of both classifiers. The results are shown in Table 4.1. In the table, “Co-Training(Bayes)” and

[1] http://www.yahoo.co.jp
[2] http://www.sanook.com, http://www.pantip.com

Table 4.1: The precision (%), recall (%) and F_1-measure of the classifiers for the problem of Thai Web page identification.

Classifier	P (%)	R (%)	F_1
ICT(Word)	100.00	99.00	99.50
S-Bayes	100.00	99.00	99.50
ICT(Bayes)	100.00	98.78	99.39
S-Word	99.08	99.61	99.34
Co-Training(Bayes)	100.00	98.67	99.33
EM	100.00	98.56	99.28
Co-Training(Word)	100.00	98.45	99.22

"Co-Training(Word)" are the results of naive Bayes and word segmentation classifiers of *Co-Training,* respectively. "ICT(Bayes)" and "ICT(Word)" are for naive Bayes and word segmentation classifiers of ICT. As shown in the table, ICT(Word) gave the best performance according to F_1-measure (99.50%), which is equal to S-Bayes. The performance of ICT (Bayes) was higher than S-Word. It can be seen from the table that the classifiers of ICT outperformed both classifiers of Co-Training as well as EM.

Compared to supervised learning 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. Though we did not include the details of running time of all classifiers, from the experiments we found that ICT ran much faster than Co-Training and EM because Co-Training and EM used incremental labeling style during the training process which gradually added a small number of labeled data in each round.

4.3 Summary

We have presented a method that effectively uses unlabeled examples to estimate the parameters of the system for Thai Web page identification. The method is based on two components, i.e. the word segmentation classifier and the naive Bayes classifier. We found that ICT is capable to boost the performance of the naive Bayes classifier.

In the following chapter, we will apply ICT to Web page categorization problems in which no domain knowledge is available.

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

CHAPTER 5

WEB PAGE CATEGORIZATION

In this chapter, we discuss the details of how we perform the various experiments and provide information on the data sets that are used in those experiments. The chapter is organized into five sections. Section 5.1 is about the feature sets used in our experiments. Section 5.2 and 5.3 are about data sets and experimental settings. The results are provided in Section 5.4 and the conclusion is shown in Section 5.5.

5.1 Feature Set

For the classification problem, the classifier's performance usually depends on the classification mechanism with the support of the feature set. An appropriate feature set will help the classifier to increase 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 words appearing in the content of a Web page and words appearing on the heading of the Web page.

5.1.1 Content

The content of a Web page provides information to the user in detail. As shown in Figure 5.1, one of the headings is written as "Honors, Awards, and Professional Service", whereas the following content is a list of activities in details. We extract all words appearing in the content to be one of our feature sets.

5.1.2 Heading

Our assumption is that a heading phrase usually represents the main idea of the following content. Thus we extract words appearing in all headings of the Web page to see the impact of this feature set.

5.1.3 Heading and Content

After the learning process is completed, the combined classifier predicts the class of examples based on the output from the heading-based and content-based classifiers. Given a set of class labels $L = \{l_1, l_2, \ldots, l_m\}$, the classifier calculates the posterior probability of each class label l_j using different feature sets as shown in Equation 5.1.

$$Pr(l_j \mid d_i) = Pr(l_j \mid x_1)\, Pr(l_j \mid x_2) \quad (5.1)$$

where x_1 and x_2 are the heading and the content feature sets of document d_i. Then the classifier predicts the class label l_j based on the maximum posterior probability.

5.2 Data Sets

In order to evaluate the ICT algorithm, we conduct various experiments on three data sets. The first data set is the WebKb (WebKb 2000), the second data set is the WebClass (WebClass 2000) and the last one is the DrugUsage data set (DrugUsage 2001).

5.2.1 WebKb

The WebKb data set contains many Web pages related to the university domain. It was obtained via ftp from Carnegie Mellon University (WebKB 2000). The data set consists of 981 Web pages collected from computer science department Web sites of four universities: Cornell, University of Washington, University of Wisconsin, and University of Texas. These Web pages are hand-labeled into four categories, which are course homepages, faculty homepages, project homepages and student homepages. We have 220 course Web pages, 147 faculty Web pages, 81 project Web pages and 533 student Web pages.

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

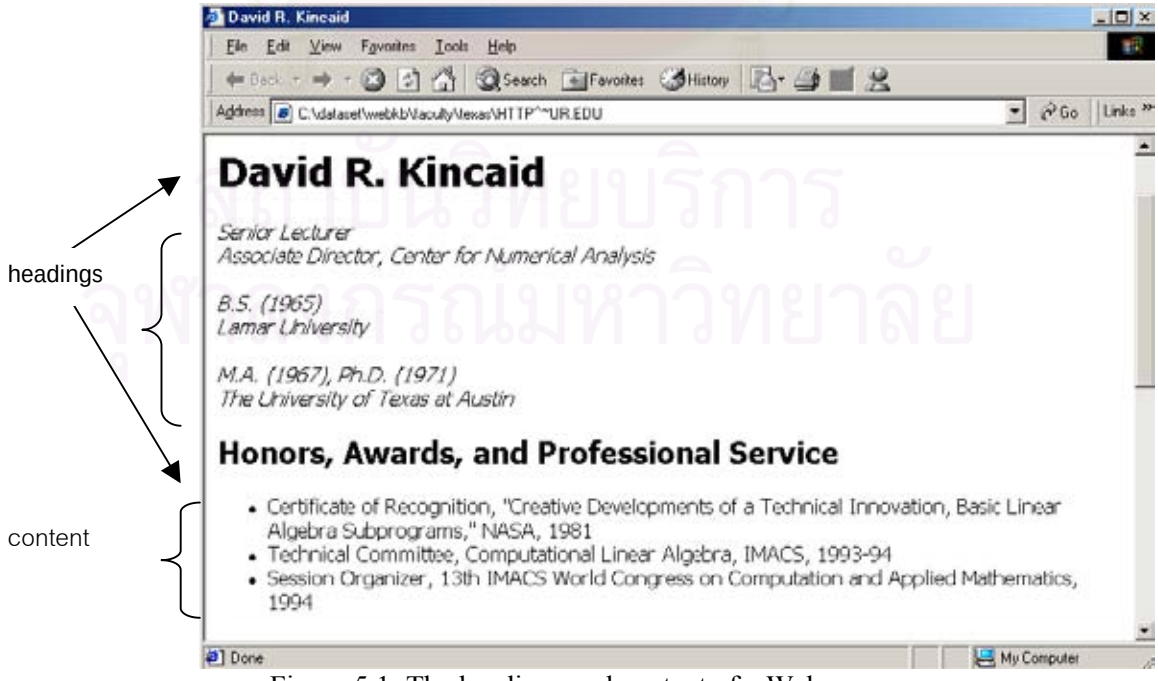


Figure 5.1: The headings and content of a Web page.

5.2.2 WebClass

The WebClass data set was obtained from a machine learning research group in Italy (WebClass 2000). This collection of Web pages provides information about the hobbies. It consists of 192 Web pages corresponding to four categories, which are astronomy, jazz, auto and motorcycle. Each category has 48 pages. The first two categories are semantically distant, whereas auto and motorcycle are both concerning about vehicles and are closely related.

5.2.3 DrugUsage

The DrugUsage data set was obtained from a research group at Sirindhorn International Institute of Technology. It consists of 353 Web pages corresponding to five categories in pharmaceutical domain. These categories are named as adverse, clinical pharmacology, overdose, patient information and warning. An adverse Web page describes about the side effects of medicines. A clinical pharmacology Web page explains about the use of medicines. An overdose Web page tells us about the symptoms of a patient when taking the drug more than usual. A patient information Web page provides information to the patient about the drug usage. A warning Web page presents the drug notification messages to the patient.

5.3 Experimental Setting

Before conducting all experiments, each Web page was preprocessed by doing html tag elimination, stop word removal and word stemming. The details of stop word removal and word stemming are provided in the following subsections.

5.3.1 Stop Word Removal

Each Web page is filtered to remove words that give no significance in predicting the class of the page. Words to be eliminated are auxiliary verb, preposition, pronoun, possessive pronoun, phone number, digit sequence, date and special character (See Appendix A for a list of stop words).

5.3.2 Word Stemming

Many different words might have a common stem. One can therefore reduce dimensionality greatly by replacing each word by its stem. The assumption is that the stem contains the important part of the meaning of the word, so stemming should be done to enhance the performance of a classifier. For example, the words "teached", "teaching", "teach", "teaches", and "teacher" all have "teach" as their stem. If we use stemming, then all

occurrences of any of these 5 words would be replaced by "teach". Stemming is usually done using a stemming algorithm and there are a great number of stemming algorithms in use (Hull 1996). In our experiment, we employed a stemming algorithm called Porter Stemming (Porter 1980).

5.4 Experimental Results

After the preprocessing step was completed, we extracted all words appearing in all headings of each Web page to be the features of the heading-based classifier and extracted all words appearing in the content to be the features of the content-based classifier. Therefore, each Web page can be viewed as the set of words appearing in the page's content and the set of words appearing in all headings. Note that all experiments were done using bag-of-words representation.

5.4.1 Experimental Results on the WebKb Data Set

The settings for the classifiers are as follows.

(1) For ICT, we randomly selected 30% of all samples from each category to be initial labeled data. The training set (unlabeled data) consisted of 30% of all samples, and 40% of all samples were used as a test set. The first classifier employed the heading feature set, whereas the second employed the content feature set.

(2) For Co-Training, we used the same set of initial labeled data as ICT. The unlabeled data and the test set were also the same. The parameters p and n were set to 1 and 3, respectively.

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

(4) For the EM classifier, we randomly selected 30% of all examples from each category to be initial labeled data. The unlabeled data and the test data consist of 30% and 40% of all examples, respectively.

The experiments were conducted using 5-fold cross-validation in order to give each Web page a chance to be trained and tested equally. After the training process was finished, we evaluated classifiers based on three feature sets, which were the heading feature set, the content feature set and the combined feature set (heading+content).

Table 5.1 shows the results of each classifier using ICT when the learning process is finished. Table 5.2 shows the result of the supervised naive Bayes classifier. Table 5.3 and 5.4 are the results of Co-Training and EM, respectively.

Table 5.1: The performance of classifiers using the ICT algorithm on the WebKb data set.

WebKb	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Course	67.69	100.00	80.73	66.15	97.73	78.90	85.71	95.45	90.32
Faculty	24.79	96.67	39.46	22.39	100.00	36.59	20.98	100.00	34.68
Project	35.00	87.50	50.06	23.26	62.50	33.90	27.03	62.50	37.74
Student	92.45	92.45	92.45	87.00	82.08	84.47	90.10	85.85	87.92
Average	71.85	94.39	78.25	67.23	86.73	71.76	73.39	88.27	76.22

Table 5.2: The performance of classifiers using the supervised naive Bayes algorithm on the WebKb data set.

WebKb	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Course	69.35	97.73	81.13	88.89	90.91	89.89	86.09	92.27	89.03
Faculty	39.22	68.97	50.00	37.93	75.86	50.57	42.84	75.49	54.58
Project	50.00	62.50	55.56	47.37	56.25	51.43	49.35	55.00	51.33
Student	90.74	91.59	91.16	87.04	87.85	87.44	90.63	86.90	88.63
Average	74.99	87.24	79.91	76.95	84.18	79.60	78.92	83.76	80.46

Table 5.3: The performance of classifiers using the Co-Training algorithm on the WebKb data set.

WebKb	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Course	68.25	97.73	80.37	89.74	79.55	84.34	85.71	95.45	90.32
Faculty	40.63	86.67	55.32	51.52	56.67	53.97	32.88	80.00	46.60
Project	24.14	87.50	37.84	25.53	75.00	38.09	37.50	18.75	25.06
Student	93.26	78.30	85.13	91.67	51.89	66.27	75.74	96.26	84.77
Average	73.95	84.69	75.64	79.69	60.72	66.14	68.29	87.26	75.30

Table 5.4: The performance of classifiers using EM algorithm on the WebKb data set.

WebKb	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Course	63.19	98.64	77.00	83.00	80.91	81.33	84.63	80.45	81.68
Faculty	26.51	90.44	40.94	35.09	89.03	48.16	28.10	97.26	42.91
Project	39.22	75.00	51.26	37.38	57.50	41.70	24.41	87.50	37.19
Student	87.23	91.58	89.20	91.95	69.89	77.05	91.68	71.02	77.91
Average	68.62	91.64	75.98	76.78	74.28	70.70	74.87	78.50	70.08

Note that we applied the micro average to measure the overall performance of each classifier. The micro average is normally used when the numbers of test data in each category are different. Considering the average of F_1 measure, we found that the heading-based classifier of ICT obtained 78.25% correctness, which was higher than those of Co-Training and EM. The content-based classifier of ICT obtained 71.76% correctness, while Co-training and EM got 66.14% and 70.70%, respectively. Nevertheless, the performance of classifiers using ICT was a bit less than those using supervised naive Bayes classifiers. This is because ICT employed only 50% of the labeled data used by the supervised naive Bayes classifier. The performances of classifiers using both feature sets of ICT were also higher than EM and Co-Training. We found that the training time of ICT was much less than Co-Training and EM. With ICT, it took about 3 minutes for the algorithm to converge, whereas with Co-Training or EM, it took more than 20 minutes to converge (all of the algorithms were implemented using JAVA language on Windows platform).

5.4.2 Experimental Results on the WebClass Data Set

The settings for classifiers are as follows.

(1) For ICT, Co-Training and EM, we selected 33% of all examples to be initial labeled data. The training set consisted of 33% and the remaining 34% was a test set.

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

Table 5.5: The performance of classifiers using the ICT algorithm on the WebClass data set.

WebClass	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Astro	93.33	100.00	96.55	100.00	92.86	96.30	100.00	92.86	96.30
Auto	60.87	100.00	75.68	93.33	100.00	96.55	76.47	92.86	83.87
Jazz	100.00	64.29	78.26	93.33	100.00	96.55	100.00	100.00	100.00
Motorcycle	91.67	78.57	84.62	93.33	100.00	96.55	93.33	100.00	96.55
Average	86.47	85.72	83.78	95.00	98.22	96.49	92.45	96.43	94.18

Table 5.6: The performance of classifiers using the supervised naive Bayes algorithm on the WebClass data set.

WebClass	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Astro	87.97	83.33	84.08	100.00	92.86	96.20	100.00	95.24	97.53
Auto	74.22	97.62	83.98	93.61	97.62	95.39	95.56	97.62	96.47
Jazz	100.00	66.67	79.34	97.78	100.00	98.85	97.78	100.00	98.85
Motorcycle	75.23	92.86	81.27	81.18	100.00	89.50	82.54	100.00	90.39
Average	84.36	85.12	82.17	93.14	97.62	94.99	93.97	98.21	95.81

Table 5.7: The performance of classifiers using Co-Training algorithm on the WebClass data set.

WebClass	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Astro	65.00	92.86	76.47	100.00	92.86	96.30	100.00	95.24	97.53
Auto	77.78	100.00	87.50	77.78	100.00	87.50	87.71	97.62	92.32
Jazz	100.00	78.57	88.04	93.33	100.00	96.55	87.78	100.00	92.97
Motorcycle	53.85	100.00	70.04	60.87	100.00	75.68	75.00	95.24	82.91
Average	74.16	92.86	80.51	83.00	98.22	89.01	87.62	97.02	91.43

Table 5.8: The performance of classifiers using EM algorithm on the WebClass data set.

WebClass	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Astro	83.45	97.62	89.51	100.00	95.24	97.53	100.00	97.62	98.77
Auto	46.13	50.00	47.94	84.26	100.00	91.39	77.31	100.00	86.94
Jazz	62.86	78.57	65.83	91.39	100.00	95.48	95.83	100.00	97.78
Motorcycle	80.00	52.38	52.72	74.73	100.00	84.86	76.11	100.00	85.35
Average	68.11	69.64	64.00	87.60	98.81	92.31	87.31	99.40	92.21

All experiments were conducted using 3-fold cross-validation. The results are shown in Tables 5.5-5.8 Considering the average performance measured by F_1, we found that the heading-based classifier of ICT obtained 83.78%, which was higher than those of Co-Training and EM.

The performance of heading-based and content-based classifiers using ICT were higher than those using S-Bayes. However, the performance deficiency of combined feature set (heading+content-based classifier) was less than those using S-Bayes. Considering the

heading-based classifier, we found that Co-Training and EM lost 2.02% and 22.11%, respectively.

For the content-based classifier, the classifier of ICT got higher performance than S-Bayes. Moreover, the classifiers of Co-Training and EM lost 6.30% and 2.81%, respectively.

5.4.3 Experimental Results on the DrugUsage Data Set

All experiments were conducted using 3-fold cross validation. The settings for classifiers were as follows.

1) For ICT, Co-Training and EM, we selected 33% from all examples to be initial labeled data. The training set consisted of 33% and the rest 34% was a test set.
2) For the supervised naive Bayes classifier, we selected 66% from all examples to be labeled data. The test set was also 34% from all examples.

Table 5.9: The performance of classifiers using the ICT algorithm on the DrugUsage data set.

DrugUsage	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Adverse	72.30	98.55	83.41	57.56	96.71	72.17	69.70	95.83	80.70
Clinical	83.33	50.72	63.06	88.28	85.82	87.03	66.67	91.67	77.20
Overdose	44.86	92.99	60.53	44.10	48.19	46.05	34.62	75.00	47.37
Patient	38.00	69.44	49.12	48.53	29.17	36.44	46.43	54.17	50.00
Warning	64.21	91.61	75.50	47.24	91.61	62.33	54.76	95.83	69.69
Average	60.54	80.66	69.17	57.14	70.30	63.04	54.44	82.50	64.99

Table 5.10: The performance of classifiers using the supervised learning algorithm on the DrugUsage data set.

DrugUsage	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Adverse	97.10	94.32	95.69	82.70	95.71	88.73	82.14	100.00	90.19
Clinical	80.86	84.42	82.60	91.54	88.71	90.10	90.91	83.33	86.96
Overdose	65.59	95.71	77.84	51.84	88.77	65.45	51.16	91.67	65.67
Patient	64.38	95.83	77.02	67.73	70.83	69.25	75.00	87.50	80.77
Warning	70.76	93.06	80.39	50.24	91.61	64.89	53.66	95.65	68.75
Average	75.74	92.67	83.35	68.81	87.12	76.89	70.57	91.63	78.47

Table 5.11: The performance of classifiers using Co-Training algorithm on the DrugUsage data set.

DrugUsage	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Adverse	78.71	88.71	83.41	59.42	91.36	72.01	67.74	91.30	77.77
Clinical	73.15	59.18	65.43	77.10	83.03	79.95	86.36	79.17	82.61
Overdose	42.40	59.18	49.40	37.00	75.91	49.75	38.89	87.50	53.85
Patient	18.06	12.50	14.77	34.50	43.06	38.30	31.25	41.67	35.72
Warning	65.21	93.06	76.69	44.24	94.44	60.25	38.33	100.00	55.42
Average	55.51	62.52	58.81	50.45	77.56	61.14	52.51	79.93	61.07

Table 5.12: The performance of classifiers using EM algorithm on the DrugUsage data set.

DrugUsage	Heading-based Classifier			Content-based Classifier			Heading+Content-based Classifier		
Category	P	R	F_1	P	R	F_1	P	R	F_1
Adverse	88.59	67.75	76.78	69.28	98.55	81.36	70.59	100.00	82.76
Clinical	70.61	33.88	45.79	67.23	83.15	74.35	68.97	83.33	75.47
Overdose	16.40	11.17	13.29	24.11	56.58	33.82	19.35	25.00	21.82
Patient	17.06	31.94	22.25	37.62	48.61	42.42	27.42	70.83	39.53
Warning	72.41	87.50	79.25	32.05	95.83	48.04	33.33	95.83	49.46
Average	53.02	46.45	49.52	46.06	76.55	57.51	43.93	75.00	55.41

Considering the average performance measured by F_1 from Table 5.9 to 5.12, we found that the heading-based classifier of ICT got 69.17% correctness which was higher than that of Co-Training and EM, which got 58.81% and 49.52%, respectively. Moreover, the content-based classifier of ICT got 63.04% which was higher than those of Co-Training and EM which got 61.14%, 57.51%. Supervised naive Bays obtained 83.35% and 76.89% correctness on heading-based and content-based classifiers. It can be seen that the ICT performance deficiency compared to supervised naive Bayes was less than those of Co-Training and EM. The heading-based and content-based classifiers of ICT lost about 17.01% and 18.01%. Nevertheless, the heading-based and content-based classifiers of Co-Training lost 29.44% and 20.48%. For the EM algorithm, the lost of performance were 40.59% and 25.20% on the heading-based classifier and the content-based classifier. For the classifiers using the combined feature set, we found that their level of performance were between the heading-based and content-based ones in every algorithm.

5.5 Summary

In this chapter, the ICT algorithm was investigated to see its impact on the Web page categorization problem. We compared ICT with Co-Training, supervised naive Bayes, EM, and finally found that ICT still outperformed the semi-supervised learning algorithm (Co-Training and EM). However, the performance of ICT was less than the supervised naive Bayes classifier on WebKb and DrugUsage data sets, since ICT employed only 50% of initial labeled data used by the supervised naive Bayes algorithm.

CHAPTER 6

EFFECT OF INITIAL LABELED DATA ON WEB PAGE CATEGORIZATION

The objectives of this chapter are to study two side effects of the ICT algorithm. The first issue is the effect of the ICT, when varying numbers of initial labeled data are supplied to the algorithm. In the Web page categorization problem, each classifier of ICT starts the learning process by estimating parameters from these initial labeled data. For this reason, different numbers of initial labeled data may effect the performance of the classifier in some ways. The second issue is the effect of the ICT algorithm in the presence of the noisy labeled data.

6.1 The Effect of the Varying Number of Initial Labeled Data

In order to see the effect of the amount of initial labeled data, the 3-fold cross validation experiments were carried on the WebClass data set. We randomly selected different proportions of training data to be initial labeled data. The size of initial labeled data was diverged from 50% to 0%. After the learning process was completed, each classifier was evaluated.

Figure 6.1-6.3 show the results of the heading-based classifiers, the content-based classifier and the combined classifiers of the learning algorithms tested in the experiments. From Figure 6.1, the heading-based classifier's performance of the ICT algorithm decreased from 83.78% to 50.00%, when the size of initial labeled data was reduced from 50% to 0%. The heading-based classifier of Co-Training got worse performance ranging from 82.00% to 47.46%. The performance of the heading-based classifier of the EM algorithm decreased from 81.44% to 37.22%. Considering the content-based features in Figure 6.2, we found that the content-based classifier's performance of the ICT algorithm was decreased from 96.49% to 51.07%, whereas those of the Co-training and the EM algorithms were decreased from 93.61% to 36.07% and from 83.80% to 24.63%, respectively.

As shown in Figure 6.3, the combined classifier of ICT obtained worse performance from 94.18% to 50.84% with the decrease of the number of initial labeled data. The performance of the other of the Co-training and EM algorithms became worse, as well. Their performances ranged from 91.49% to 38.33% and 81.13% to 30.43%, respectively. Note that all experimental results of the other data sets, the WebKb and the DrugUsage, showed similar results as the WebClass data set. All classifiers' performance increased, when they obtained more initial labeled data.

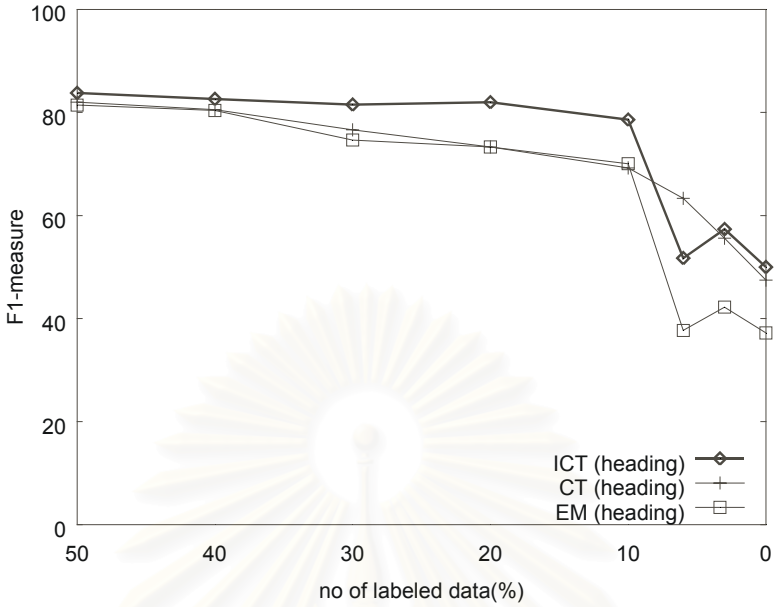


Figure 6.1: The performances of heading-based classifiers using varying numbers of initial labeled data on the WebClass data set.

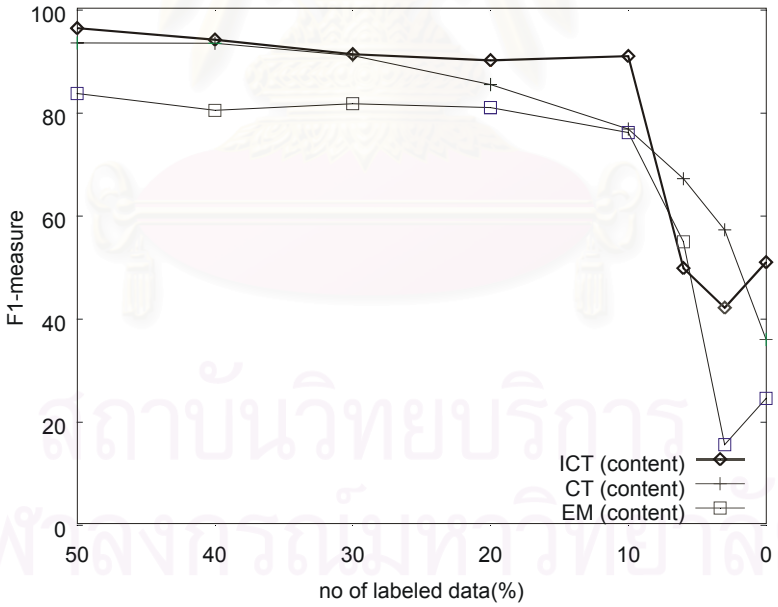


Figure 6.2: The performances of content-based classifiers using varying numbers of initial labeled data on the WebClass data set.

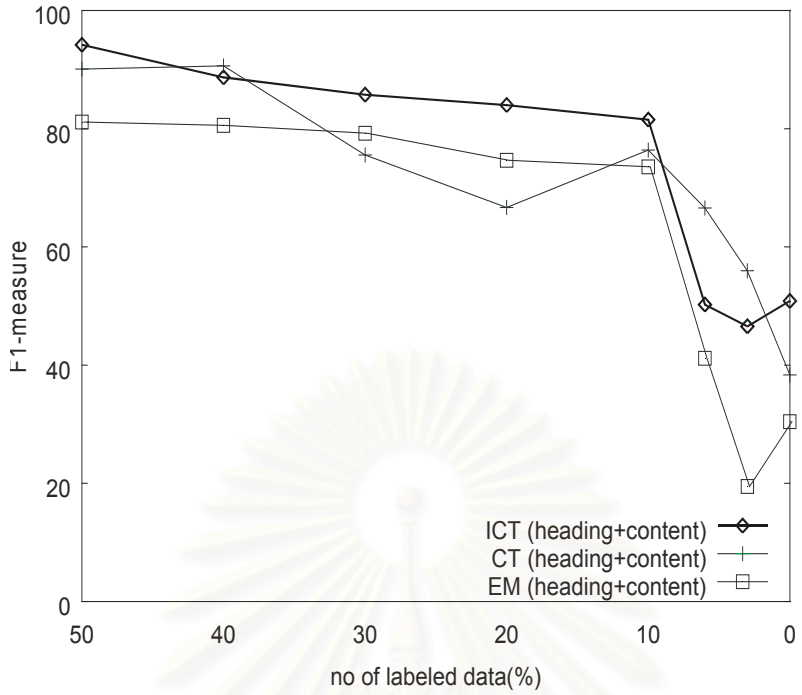


Figure 6.3: The performances of heading+content-based classifiers using varying numbers of initial labeled data on the WebClass data set.

6.2 The Effect of Noisy Labeled Data

In reality, there is a possibility that the initial labeled data are incorrectly labeled due to human error. Therefore, it is interesting to see how the learning algorithm is affected by this real world problem.

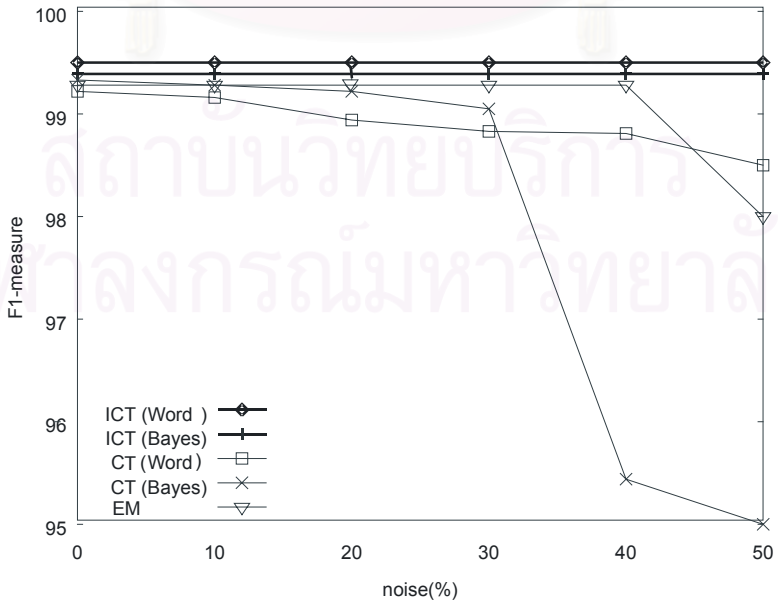


Figure 6.4: The performance of classifiers on the Thai/non-Thai data set.

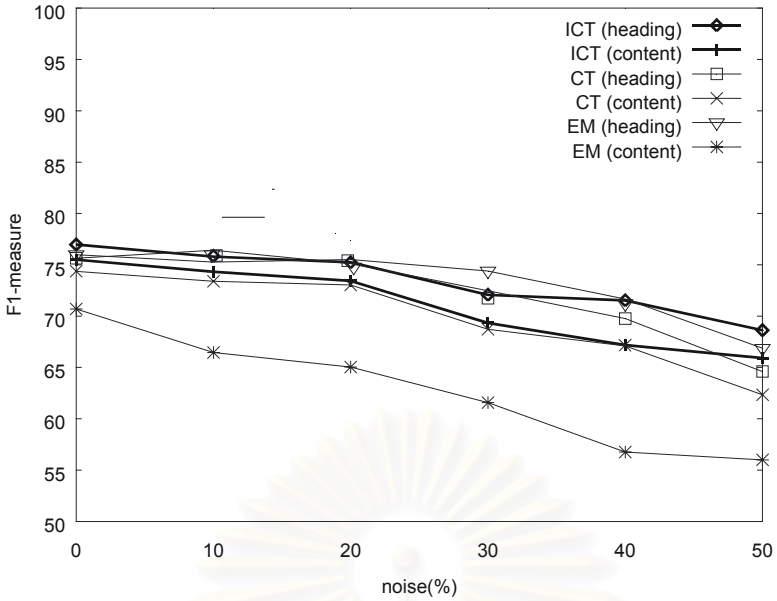


Figure 6.5: The performance of classifiers on the WebKb data set.

Since ICT, Co-Training and EM are boosting-style learning algorithms, they need a small amount of initial labeled. Therefore, we added noise to these labeled data and let the algorithms start the learning process. The experiments were conducted on three data sets as in the previous subsections. The varying levels of random class noise, between 10% to 50% were added to the initial labeled data. The results are shown below.

As shown in Figure 6.4, ICT is robust in the presence of noise as neither classifier's performance change. The word segmentation and naive Bayes classifiers of ICT still preserved their performance at 99.50% and 99.39%, when noise was added up to 50%. This was because all unlabeled data were relabeled in every iterations by both classifiers. Both classifiers, CT (Word) and CT (Bayes), of Co-Training algorithm were sensitive to noise as their performances dropped considerably from 99.22% to 98.50%. and from 99.33% to 95.00%. The reason that Co-Training was more sensitive is that it used an incremental labeling style. The noisy initial labeled data had very high influence to each classifier of Co-Training. Since the classifiers learned from the incorrectly labeled data would very likely assign the new wrong labels incrementally to the unlabeled data. These new mislabeled data would be accumulated during the training process, which caused the performance degradation of the Co-Training. The performance of the EM algorithm dropped slightly when noise was increased to 50%.

The graphs in Figure 6.5 shows the performances of all classifiers on the WebKb data set. We found that, when noise was added up to 50%, the heading-based and content-based classifiers of ICT lost about 10.85% and 12.70%, respectively. Both classifiers of Co-Training lost 14.58% and 16.17% of performance. Considering the EM algorithm, we found

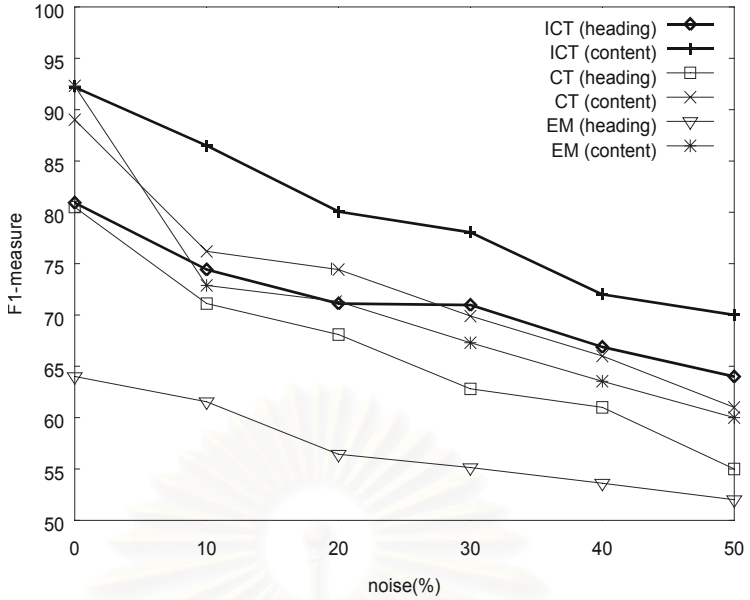


Figure 6.6: The performance of classifiers on the WebClass data set.

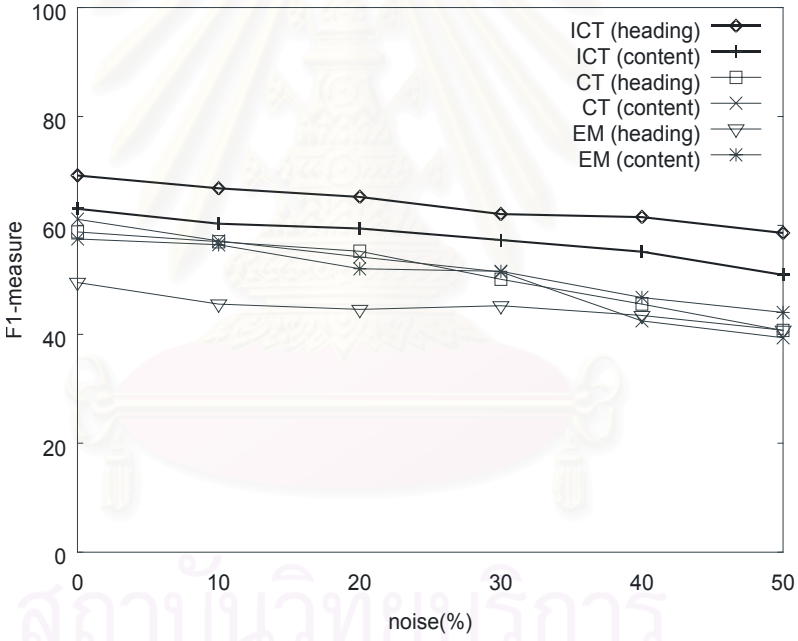


Figure 6.7: The performance of classifiers on the DrugUsage data set.

that the loss of performance due to noise labeled data is still acceptable. The heading-based classifier of EM lost 11.99% of correctness, which was comparable to ICT. The content-based classifier of EM lost 20.79%.

The performance of all classifiers, when noise was added on the WebClass data set are shown in Figure 6.6. We found that both classifiers of ICT lost less performance than those of Co-Training. The performance loss of heading-based and content-based classifiers of ICT was 20.92% and 24.1%, whereas the heading-based and content-based classifiers of

Co-Training lost 31.70% and 31.47%. For the EM algorithm, the heading-based and content-based classifiers lost 18.75% and 35%, respectively.

Considering the experimental results on the DrugUsage data set (see Figure 6.7), we found that the loss of both classifiers of ICT's performance were less than those of Co-Training. The heading-based and content-based classifiers of ICT lost 15.25% and 19.24% of F_1-measure, whereas those of Co-Training lost 30.78% and 35.62%, respectively. The content-based and heading-based of EM lost 17.71% and 23.49%.

6.3 Summary

In this chapter, we focused on the behaviors of ICT in two aspects. The first aspect concerns on the different numbers of initial labeled data. We found that the performance of ICT was enhanced, when the initial labeled data was enlarged. This is true for all data sets, which can be seen from the experimental results. The reason that the amount of initial labeled data plays an important role during the learning process is because in each round of ICT the active classifier learns from this initial labeled data plus the new label data acquired from the other classifier. The larger amount of initial labeled data dominates the learner resulting to a better performance. This can be explained in another viewpoint. Considering the machine learning paradigm, a supervised learning algorithm usually outperforms a semi-supervised and unsupervised learning algorithm. Since the supervised learning algorithm obtains all of labeled training data, whereas the semi-supervised learning receives only a partial amount of labeled data. The second point considered in this chapter is about the noisy labeled data. We found that the performance loss of both classifiers in ICT were less than those of EM and Co-Training.

CHAPTER 7

COMBINING INDUCTIVE LOGIC PROGRAMMING WITH ICT

In this chapter, we present the way how to apply Inductive Logic Programming (ILP) in ICT. The chapter is organized into four sections. The first section provides an introduction to ILP. Section 7.2 provides the detail of how to apply ILP to the Web page categorization problem. Section 7.3 gives detail about the experimental settings and results. Finally, the summary of applying ILP is given in Section 7.4.

7.1 Introduction to ILP

Inductive Logic Programming (ILP) is a research area formed at the intersection of Machine Learning and Logic Programming (Muggleton 1991). An ILP system learns a set of rules to represent the target function. The learned rules can be used to classify examples by putting them into a logic programming language, such as Prolog. That means ILP employs techniques from both machine learning and logic programming. ILP systems develop predicate descriptions from examples and background knowledge. The examples, background knowledge and final descriptions are all described as logic programs. The learning mechanism involves a unifying theory of Inductive Logic Programming that is being built up around lattice-based concepts such as refinement, least general generalization and inverse resolution in order to produce the disjunctive set of rules.

7.1.1 The Framework of ILP

In this section, we first consider how the ILP system learns a rule set. Then a number of important Inductive Logic Programming systems are explored, such as GOLEM (Muggleton and Feng 1990), FOIL (Quinlan 1990) and PROGOL (Muggleton 1995).

The basic algorithm called *sequential covering* is the most widely used approach to learn disjunctive set of rules. As shown in Table 7.1, the concept of the sequential covering algorithm is to sequentially learn a single rule by calling the subroutine, *learn-one-rule*, and accumulate the learned rule into the rule set. After the positive examples covered by the learned rule are removed, the subroutine is called again to produce the next rule from the remaining examples. The learn-one-rule subroutine accepts a set of positive and negative training examples as input, then outputs a single rule that covers many of the positive examples and few of the negative examples. The characteristic of an output rule is that it should have high accuracy, but not necessarily to have high coverage. One strategy to implement the learn-one-rule subroutine is to do a general-to-specific search through the space of possible rules to search for the high accuracy rule.

Table 7.1: The sequential covering algorithm.

Sequential-Covering (*Target_attribute*, *Attributes*, *Examples*, *Threshold*)

- *Learned_rules* ← {}
- *Rule* ← Learn-one-rule (*Target_attribute*, *Attributes*, *Examples*)
- while Performance (*Rule*, *Examples*) > *Threshold*, do
    - *Learned_rules* ← *Learned_rules* + *Rule*
    - *Examples* ← *Examples* −{examples correctly classified by *Rule*}
    - *Rule* ← Learn-one-rule (*Target_attribute*, *Attributes*, *Examples*)
- *Learned_rules* ← sort *Learned_rules* according to Performance over *Examples*
- Return *Learned_rules*

The search style is a greedy depth-first search with no backtracking. This method can be improved by performing a beam search to relax the suboptimal problem of depth-first search. As illustrated in Table 7.2, the algorithm keeps track of *k* best candidate_hypotheses at each step. The rule specialization is processed for each candidate.

In Table 7.2, *candidate_hypothesis* is the conjunction of attribute-values or predicates. Each of these conjunctive hypotheses is a candidate set of preconditions for the rule to be learned and is evaluated by the *entropy* (see Equation 7.1) of the examples it covers. The search considers increasingly specific candidate hypotheses until it reaches a maximally specific hypothesis that contains all available attributes. The algorithm outputs the best rule that has the highest performance.

$$-Entropy(S) = \sum_{i=1}^{c} p_i \log_2 p_i \tag{7.1}$$

Where c is the number of distinct classes of examples. P_i is the proportion of examples from S for which the target function takes on the ith value.

7.1.2 ILP Systems

Many ILP systems have been developed such as GOLEM, FOIL, PROGOL, etc. In this section, we describe the techniques of these systems.

The GOLEM system was launched in 1990 by Muggleton and Feng (1990). The system conducts a *specific-to-general* search. The specific hypothesis is first created by randomly selecting several positive examples. The process of GOLEM is based on a technique called inverse resolution, which corresponds to the relative least-general generalizations, *rlgg* operator of Plotkin (1971).

Table 7.2: The learn-one-rule algorithm.

Learn-one-rule (*Target_attribute*, *Attributes*, *Examples*, *k*)

- Initialize *Best_hypothesis* to the most general hypothesis
- Initialize *Candidate_hypotheses* to the set {Best_hypothesis}
- While *Candidate_hypotheses* is not empty, Do
  1. Generate the next more specific candidate_hypotheses
     - *All_constraints* ← the set of all constraints of the form ($a = v$), where a is a member of Attributes, and v is a value of a that occurs in the current set of Examples
     - *New_candidate_hypotheses* ←
       For each *h* in Candidate_hypotheses,
       For each *c* in All_constraints,
       Create a specialization of *h* by adding the constraint *c*
     - Remove from *new_candidate_hypotheses* any hypotheses that are duplicates, in consistent, or not maximally specific
  2. Update *Best_hypothesis*
     - For all *h* in *new_candidate_hypotheses* do
       If (performance (*h*, *Examples*, *Target_attribute*)
       > performance(*Best_hypothesis*, *Examples*, *Target_attribute*))
       then *Best_hypothesis* ← *h*
  3. Update *Candidate_hypotheses*
     - *Candidate_hypotheses* ← the *k* best member of *New_candidate_hypotheses*, according to the performance measure.
- Return a rule

The FOIL system (Quinlan 1990) employs a sequential-covering algorithm. It learns one rule at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule (Mitchell 1997). The system searches (using hill-climbing technique) "*general-to-specific*", by starting with the most general hypothesis and adding one literal at a time to specialize the rule until it avoids all negative examples. During the search, the system also tries to keep the remaining hypotheses sufficient enough to be the representative of the positive examples.

The PROGOL system (Muggleton 1995) uses a technique called inverse entailment to generate the single most specific hypothesis that, together with the background information,

entails the observed data (Mitchell 1997). PROGOL uses the sequential covering algorithm to learn a set of rules from hypothesis space. The system performs *general-to-specific* search of the hypothesis space. It employs A* search along the way to find a set of rules that represent the concept of the class. The detail of PROGOL are described by Muggleton (1995). Many researchers point out that PROGOL is seen as a standard ILP learner and is often used as a benchmark when new ILP systems are introduced. He also points to previous successes using PROGOL of such as protein shape prediction by Muggleton et al. (1992) and Drug Design by Finn et al. (1998).

7.2 Learning the Concept of Web Pages with ILP

Since ICT consists of two learners, a strong and a weak learner, we embed an ILP system to be the strong learner and the naive Bayes to be a weak learner (as shown in Figure 7.1). In this section, we present our ILP framework in order to solve the Web page categorization problem. We employ PROGOL as the strong learner and make use of the induced rules to classify training examples (*TrainingData2*).

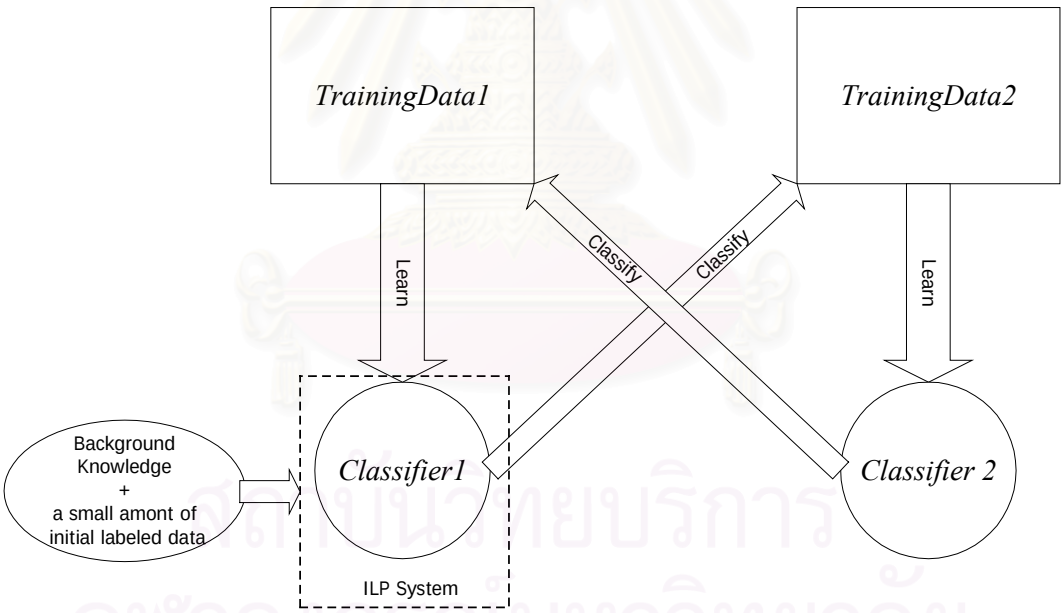


Figure 7.1 Combining an ILP system with ICT.

As shown in Figure 7.1, the ILP system is supplied with two set of examples, i.e., a small amount of initial labeled data and unlabeled data. The ILP system makes use of background knowledge about the categories of Web pages together with a set of initial labeled data to induce a set of rules. Then the system classifies unlabeled examples using the rule set and feeds the newly labeled examples to the naive Bayes (*Classifier2*) to learn and classify *TrainingData1* into categories. The ILP system then uses the background knowledge

and *TrainingData1* as labeled examples to induce a new rule set. This process is repeated until the system is converged.

7.2.1 Feature Sets

After the preprocessing step (stop word removal and word stemming) is accomplished, the feature extraction process is started. For the ILP system, the feature sets of each Web page are constructed in predicate forms. We extract three feature sets from each Web page as follows.

- A title predicate, has_title(*p*,*word*), is created using words appearing in the title of the Web page, *p*.
- A heading predicate, has_head(*p*,*word*), is created using words appearing in all headings of the Web page, *p*.
- A hyperlink predicate, has_link(*p*,*word*), is created using words appearing in all hyperlinks of the Web page, *p*.

7.2.2 Background Knowledge

The domain knowledge is an important part of an ILP system. The use of domain knowledge is essential for achieving the intelligent behavior. We supply the ILP system with background knowledge for each Web page category. This knowledge is written in predicate forms. Below is an example list of background knowledge of a course homepage, and the complete list of background knowledge used in the following experiment is given in Appendix B.

classMaterial(textbook).
classMaterial(slide).
classMaterial(syllabus).
assignment(project).
assignment(homework).

After the learning process is completed, one possible induced rule is as follows.

coursehomepage(*A*) :- has_link(*A*,*B*) , *assignment*(*B*).

It means, a Web page, *A*, which can be classified as a course homepage must have word *B* which is defined by predicate *assignment* in the background knowledge. The explicit explanation is that the Web page must have the word “project” or “homework” appeared in its hyperlinks.

7.3 Experimental Results

We embed the ILP system (PROGOL) in our ICT algorithm in order to see the performance enhancement of the Web page categorization problem, since the previous experimental results (using naive Bayes in both learners) were still not high, due to the fact that the naive Bayes learners are not strong enough to produce the satisfactory performance. In this section, the experimental results on the WebKb and the DrugUsage data sets are reported.

7.3.1 Experimental Results on the WebKb Data Set

For ICT, Co-Training and EM, we randomly selected 30% of all examples from each category to be initial labeled data. The unlabeled training data consisted of 30% of all examples, and 40% of all examples were used as a test set. The experiments were conducted using 5-fold cross-validation. Table 7.3 shows the results of all experiments conducted on the WebKb data set. In the table, ICT-ILP stands for the performance of ICT which combines the ILP system in one of the classifiers. ICT-NB is ICT which combines two naive Bayes classifers, each of which learns from different feature sets. Co-Training stands for the Co-Training algorithm, S-Bayes stands for the supervised naive Bayes algorithm. Note that the *Classifier1* of ICT-ILP is the *Progol* system. For other algorithms, *Classifier1* means the heading-based classifier. The *Classifier2* is the content-based classifier for all of the algorithms.

Considering the two versions of ICT (ICT-ILP and ICT-NB) in Table 7.3, we found that ICT-ILP's performance measured by F_1 was increased from 78.25% to 80.90% on *Classifier1*. Moreover, *Classifier1* was able to boost the performance of *Classifier2*. The *Classifier2*'s performance was enhanced from 71.76% to 84.44%. Compared to the supervised naive Bayes algorithm, ICT-ILP outperformed S-Bayes on *Classifier1*. The reason that ICT-ILP got the highest performance came from the contribution of the strong learner (the Progol system).

The learning process of ICT-ILP took more time than the ICT-NB, since the strong learner, Progol, needed a long time to do a general-to-specific search to get the optimum set of rules. In each iteration of ICT, the Progol took about 1 hour to generate the rules, therefore it took 3 hours for ICT-ILP to converge.

Table7.3: The average performance of classifiers on the WebKb data set.

	Classifier1			*Classifier2*		
Algorithm	P	R	F_1	P	R	F_1
ICT-ILP	80.00	81.82	80.90	82.61	86.36	84.44
ICT-NB	71.85	94.39	78.25	67.23	86.73	71.76
Co-Training	73.95	84.69	75.64	79.69	60.72	66.14
S-Bayes	74.99	87.24	79.91	76.95	84.18	79.60
EM	68.62	91.64	75.98	76.78	74.28	70.70

Table 7.4: The average performance of classifiers on the DrugUsage data set.

	Classifier1			*Classifier2*		
Algorithm	P	R	F_1	P	R	F_1
ICT-ILP	82.37	98.32	89.90	56.03	88.20	65.39
ICT-NB	60.54	80.66	69.17	57.14	70.30	63.04
Co-Training	55.51	62.52	58.81	50.45	77.56	61.14
S-Bayes	75.74	92.67	83.35	68.81	87.12	76.89
EM	72.41	87.50	79.25	33.33	95.83	49.46

7.3.2 Experimental Results on the DrugUsage Data Set

For ICT, Co-Training and EM, we selected 33% of all examples to be initial labeled data. The training set consisted of 33% and the remaining 34% was a test set. For the supervised naive Bayes classifier, we selected 66% of all examples to be labeled data. The test set consisted of 34% of all examples. All experiments were conducted using 3-fold cross validation.

For the performance of *Classifier1* (as shown in Table 7.4), ICT-ILP got the highest F_1. ICT-NB's performance was increased from 69.17% to 89.90%. This means that the ILP system had contributed 30% of performance enhancement to ICT-NB. For *Classifier2*, ICT-ILP got 65.39% measured by F_1, which was higher than that of ICT-NB. The overall learning process of ICT-ILP took 1 hour to converge.

7.4 Summary

In this chapter, we have presented the enhancement version of ICT using the ILP system. We found that the induced rules have more efficiency in classify unlabeled examples. This evidence can be seen from all experimental results. The benefit of the ILP system can be seen clearly when all categories in the data set are closely related. The reason is that most of the words in the closely related categories are likely to be equally distributed. Thus using the

statistical approach like a naive Bayes learner might not perform well enough to distinguish the difference between categories. The representation of the rule sets, on the other hand, can point out the specific location in each Web page that can be used as a standard prototype of the categories.

CHAPTER 8

SUMMARY AND FUTURE WORKS

8.1 Summary

The amount of available information on the Web is increasing rapidly. Users tend to rely on search engines to find the information they are looking for. However, finding relevant information using the search engines often provides unsatisfactory results. One approach to solve this problem is to manage all Web pages into categories, which is our main concern in this thesis. The motivation of this research was to develop an algorithm that has an ability to categorize Web pages.

In this thesis, we have presented a novel Web page categorization algorithm called Iterative Cross-Training (ICT). ICT consists of two learners, the strong and the weak learners, each of which learns from different feature sets of training data and uses its knowledge to classify the training data of the other learner into categories. The learning process is done in crossing style.

Several characteristics make ICT a promising algorithm for the Web page categorization. First, it is collaborated, employing two learners to help each other in acquiring the knowledge. Second, it is semi-supervised. It requires only a small amount of initial labeled data but can provide an acceptable performance.

We have evaluated the ICT algorithm and compared it to other learning algorithms. We used four data collections in our experiment: Thai/non-Thai data set for the Thai Web page identification problem; and WebKb, WebClass and DrugUsage for the Web page categorization problem. Our experiments demonstrated that the ICT algorithm produced the high quality of results.

We have proceeded to investigate the behavior of ICT, when different numbers of initial labeled data were supplied to the algorithm. We found that ICT's performance was increased as the numbers of initial labeled data were increased. Furthermore, the performance of ICT in the presence of noisy labeled data was analyzed. We found that the ICT algorithm was robust to noise when domain knowledge was available to the algorithm (in the problem of Thai Web page identification when we supplied ICT with the knowledge in the form of Thai dictionary). In the case that no domain knowledge are available (in the problem of Web page categorization), the performance loss of ICT was less than the other algorithms.

After evaluating the ICT algorithm, we turned our attention to the study how to enhance the performance of the ICT. An *Inductive Logic Programming* paradigm was introduced to the ICT. We employed the *Progol* system to be the strong learner of ICT. The experimental results showed that the ICT's performance was improved. Our results shows

that the ICT algorithm is a high potential algorithm. Its performance can be boosted by the strong learner. The weak learner becomes stronger and can be applied in the real world applications, since its computational time is cheaper and more practical.

From the real-time applications point of view, the weak learner seems to be more suitable than the strong learner. The strong learner, on the other hand, can be applied in the applications that have no time constraint, since the strong learner takes more computational time than the weak one. Furthermore, the ICT algorithm can be used in the applications but its computational time is more than each separated learner.

8.2 Future Works

In this section, we present some directions for the future work.

8.2.1 The Variant of ICT

As stated previously that the ICT algorithm is a collaborated algorithm, therefore it is interesting to study the performance when ICT consists of different numbers of learners. In which way should these learners cooperate each other during the learning process is another question that is of our interest.

8.2.2 Theoretical Analysis of ICT

In this thesis, we conducted various experiments to convince that ICT was a promising algorithm. Nevertheless, the theoretical analysis should be done to fulfill the proof of the ICT algorithm.

8.2.3 The Strong Learner of ICT

After investigation the performance of the ICT algorithm, we found that the strong learner plays an important role on the performance of the ICT algorithm. Therefore, we introduced to the ICT algorithm, the ILP system, which is considered to be stronger than the naive Bayes learner. The capability of the ILP system can be enhanced by two factors as follows:

1) The inductive bias

   There are two types of inductive bias (Lavrac 1994) : syntactic bias and semantic bias. The syntactic bias relates to the declarative language of the rules. The Progol system constructs the rules in *if-then* format. The system would be more powerful, if it can learn to construct the rules in various formats. The semantic bias concerns on the heuristic approach, the search strategy and the stopping criteria. The study on these issues should be able to enhance the performance of the ILP system.

2) The background knowledge

Since the quality of learned rules in the ILP is influenced by the background knowledge provided by the user. Therefore, a suitable set of background knowledge should be studied prior to provide to the system. We believe that the appropriate background knowledge will provide the ICT with the high potential result.

REFERENCES

Aha, D.W. 1997. Feature weighting for lazy learning algorithms. AI Review Vol. 11, pp. 1-20.

Apte, A.; Damerau, F.; and Weiss, S. 1998. Text mining with decision rule and decision Trees. In Proc. Conf. Automated Learning and Discovery, workshop, pp. 487-499.

Blum, A. and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. In Proc. of the 11th Annual Conf. Computational Learning Theory.

Breiman, L. 1996. Bagging predictors. Machine Learning 24: 123-140.

Brin, S. 1998. Extracting patterns and relations from the World Wide Web. In Proc. of the WebDB Workshop at EDBT.

Chen, H. and Kam, H. T. 2000. Evaluation of decision forests on text categorization, In Proc. of the 7th Conf. Document Recognition and Retrieval, pp. 191-199.

Cohen, W. 1996. Learning with set-valued features. In Proc. of the 13th National Conf. Artificial Intelligence, Portland, Oregon.

Cohen, W. and Singer, Y. 1998. Context-sensitive learning methods for text categorization. ACM Transactions on Information System 17(2): 141-173.

Craven M.; Slattery, S.; and Nigam, K. 1998. First-order learning for Web mining. In Proc. of the 10th European Conf. Machine Learning, pp. 250-255.

Dempster, A.P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society Series B 39:1-38.

Domingos, P. and Pazzani, M. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning 29:103.

Drucker, H.; Vapnik, V.; and Wu, D. 1999. Automatic text categorization and its applications to text retrieval. IEEE Transactions on Neural Networks 10(5): 1048-1054.

DrugUsage Data set [Online]. Available from: http://www.kindcu.siit.ac.th [2001, June 21]

Dubes, R. and Jain, A. 1988. Algorithms for clustering data. Prentice Hall.

Dumais, S. T. and Chen, H. 2000. Hierarchical classification of Web content. In Proc.of the 23rd ACM Int. Conf. on Research and Development in Information Retrieval, pp. 256-263.

Dumais, S. T.; Platt, J.; Heckerman, D.; and Sahami, M. 1998. Inductive learning algorithms and representations for text categorization. In Proc. of the 7th ACM Int. Conf. on Information and Knowledge Management, pp. 148-155.

Finn, P.; Muggleton, S.; Page, D.; and Srinivasan, A. 1998. Pharmacophore discovery using the Inductive Logic Programming system Progol. Machine Learning 30: 241-270.

Freund, Y. and Schapire, R. 1996. Experiments with a new boosting algorithm. In Proc. of the Int. Machine Learning Conf., pp. 148-156.

Han, Eui-Hong; Karypis, G.; and Kumar, V. 2001. Text categorization using weight adjusted k-Nearest Neighbor classification. In Proc. of the Pacific-Asia Conf. Knowledge Discovery and Data Mining, pp. 53-65.

Hull, D. 1996. Stemming algorithms: a case study for detailed evaluation, Journal of the American Society for Information Science 47(1): 70-84.

Joachims, T. 1998. Text categorization with Support Vector Machines: learning with many relevant features, In Proc. of the European Conference on Machine Learning.

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.

King R. and Sternberg, M.J.E. 1990. A machine learning approach for the prediction of protein secondary structure. Journal of Molecular Biology 216:441-457.

Klinkenberg, R. and Joachims, T. 2000. Detecting concept drift with Support Vector machines. In Proc. ICML-00, 17th Int. Conf. on Machine Learning, pp. 487-494.

Lavrac, N. and Dzeroski, S. 1994. Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York.

Lewis, D.D. 1999. Naive (Bays) at forty: The independence assumption in information retrieval, In Proc. of Int. Conf. Machine Learning.

McCallum, A. and Nigam, K. 1998. Employing EM and pool-based active learning for text classification. In Proc. of 15th Int. Conf. Machine Learning, pp. 350-358.

McCallum, A.; Rosenfeld, R.; Mitchell, T.; and Nigam, A. 1998. Improving text classification by shrinkage in a hierarchy of classes. In Proc. the 15th Int. Conf. Machine Learning, pp. 350-358.

Meknavin, S.; Charoenpornsawat, P.; and Kijsirikul, B. 1997. Feature-based Thai word segmentation. In Proc. Natural Language Processing Pacific Rim Symposium '97.

Mitchell, T. M. 1997. Machine Learning. 180-184, McGraw-Hill. New York.

Muggleton, S. and Feng, C. 1990. Efficient induction of logic programs. In Proc. of the 1st Conf. Algorithmic Learning Theory.

Muggleton, S. 1991. Inductive Logic Programming. New Generation Computing 8(4): 295-318.

Muggleton, S.; King, R.; and Sternberg, M. 1992. Protein secondary structure prediction using logic-based machine learning. Protein Engineering, 5(7).

Muggleton, S. 1995. Inverse entailment and progol. New Generation Computing 13:245-286.

Nigam, K.; McCallum, A.; Thrun, S.; and Mitchell, T. 1999. Text classification from labeled and unlabeled documents using EM. Machine Learning 39(2):103-134.

Pazzani, M. 1997. Learning and revising user profiles: the identification of interesting Web sites. Machine Learning 27(3): 313-331.

Pierre, J.M. 2000. Practical issues for automated categorization of Web sites. In Proc. of Conf. Sematic Web.

Plotkin, G.D. 1971. Automatic methods of inductive inference. Ph.D. Thesis, Edinburgh University.

Porter, M.F. 1980. An algorithm for suffix stripping. Program 14(3): 130-137.

Quinlan, J.R. 1990. Learning logical definitions from relations. Machine Learning 5(3):239-266.

Rocchio, J. 1971. Relevance feedback information retrieval. The Smart retrieval system-experiments in automatic document processing, pp. 313-323. Prentice-Hall.

Sébillot, P.; Bouillon, P.; and Fabre, C. 2000. Inductive Logic Programming for corpus-based acquisition of semantic lexicon. In Proc. of the 4th Conf. Computational Natural Language Learning and the Second Learning Language in Logic Workshop, pp. 199-208.

Srinivasan, A. and King, R.D. 1999. Feature construction with Inductive Logic Programming: a study of quantitative predictions of biological activity aids by structural attributes. Data Mining and Knowledge Discovery 3(1): 37-57.

Srinivasan, A.; Muggleton, S.; King R.D.; and Sternberg M.J.E. 1996. Theories for mutagenicity: a study of first-order and feature based induction. Artificial Intelligence 85:277-299.

Taira, H. and Haruno, M. 1999. Feature selection in SVM text categorization. In Proc. of the 16th Conf. of the American Association for Artificial Intelligence, pp. 480-486.

van Rijsbergen, C.J. 1979. Information Retrieval. Butterworths, London.

Vapnik, V.N. 1995. The Nature of Statistical Learning Theory. Springer, New York.

WebClass Data set [Online]. Available from: http://www.di.uniba.it/~malerba/software/webclass/webclass.html[2000, September 1]

WebKb Data set [Online]. Available from: http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-51/www/co-training/data/course-cotrain-data.tar.gz. [2000, June 14]

Weiss, S. and Indurkhya, N. 1993. Optimized rule induction. IEEE EXPERT 8(6):61-69.

Witten, I. and Frank, M. 2000. Data mining: practical machine learning tool and technique with Java implementation, Morgan Kaufmann, San Francisco, 2000.

Yang, Y. 1998. An evaluation of statistical approaches to text categorization. Journal of Information Retrieval 1(1): 67-88.

Yang, Y. and Liu, X. 1999. A re-examinationof text categorization methods. In Proc. of the 22nd ACM Int. Conf. Research and Development in Information Retrieval, pp. 42-49.

Yang, Y. and Pederson, J. 1997. Feature selection in statistical learning of text categorization. In Proc. of the 14th Int. Conf. Machine Learning, pp. 412-420.

Zamir, O. and Etzioni, O. 1998. Web document clustering: a feasibility demonstration, Research and Development in Information Retrieval, 46-54.

APPENDICES

APPENDIX A

STOP WORDS LIST

a
about
above
across
after
afterwards
again
against
all
almost
alone
along
already
also
although
always
am
among
amongst
amoungst
amount
an
and
another
any
anyhow
anyone
anything
anyway
anywhere
are
around
as
at
back
be
became
because
become
becomes
becoming
been
before
beforehand
behind
being
below
beside
besides
between
beyond
bill
both
bottom
but
by
call
can
cannot
cant
co
computer
con
could
couldnt
cry
de
describe
detail
do
done
down
due
during
each
eg
eight
either
eleven
else
elsewhere
empty
enough
etc
even
ever
every
everyone
everything
everywhere
except
few
fifteen
fify
fill
find
fire
first
five
for
former
formerly
forty
found
four
from
front
full
further
get
give
go
had
has
hasnt
have
he
hence
her
here
hereafter
hereby
herein
hereupon
hers
herself
him
himself
his
how
however
hundred
i
ii
iii
iv
vi
vii
viii
x
ix
ie
if
in
inc
indeed
interest
into
is
it
its
itself
keep
last
latter
latterly
least
less
like
ltd
made
many
may
me
meanwhile
might
mill
mine
more
moreover
most
mostly
move
much
must
name
namely
neither
never
nevertheless
next
nine
no
nobody
none
noone
nor
not
nothing
now
nowhere

of
off
often
on
once
one
only
onto
or
other
others
otherwise
our
ours
ourselves
out
over
own
part
per
perhaps
please
put
rather
re
same
see
seem
seemed
seeming
seems
serious
several
she
should
show
side
since
sincere
six
sixty
so
some
somehow
someone
something
sometime
sometimes
somewhere
still
such
system
take

ten
than
that
the
their
them
themselves
then
thence
there
thereafter
thereby
therefore
therein
thereupon
these
they
thick
thin
third
this
those
though
three
through
throughout
thru
thus
to
together
too
top
toward
towards
twelve
twenty
two
un
use
using
usually
under
until
up
upon
us
very
via
was
we
well
were
what

whatever
when
whence
whenever
where
whereafter
whereas
whereby
wherein
whereupon
wherever
whether
which
while
whither
who
whoever
whole
whom
whose
why
will
with
within
without
would
yet
you
your
yours
yourself
yourselves
jan
january
feb
february
mar
march
apr
april
may
jun
june
july
aug
august
sept
september
oct
october
nov
november
dec

december
mon
monday
tue
tuesday
wed
wednesday
thursday
fri
friday
sat
saturday
sun
sunday
home
click
here
next
just
near
far
new
newly
newest
old
see
saw
end
need
move
i
or
ok
it
the
go
went
gone
send
sent
es
do
thank
told
took
tell
take
done
did
get
got
high
low

sure	small	up	has
begin	there	down	stop
began	here	left	start
begun	meet	right	want
were	what	show	now
them	why	come	www
seen	where	came	homepage
that	when	the	co
let	which	ask	http
know	who	for	com
knew	whom	from	gif
good	whose	of	jpeg
bad	short	have	late
big	long	had	pm

APPENDIX B

LIST OF BACKGROUND KNOWLEDGE

B 1 Course Homepage

subject(cs)
subject(cse)
subject(ee)
assignment(assign)
assignment(solution)
assignment(homework)
assignment(problem)
assignment(question)
assignment(quiz)
class(class)
class(cours)
class(lectur)
class(lab)
class(prerequisit)
class(teach)
semester(fall)
semester(winter)
semester(spring)
semester(autumn)
material(handout)
material(syllabu)
material(textbook)
material(book)
material(sheet)
material(materi)
exam(midterm)
exam(final)
exam(exam)
exam(grade)

B 2 Student Homepage

sport(soccer)
sport(hockei)
sport(fish)
sport(music)
sport(basebal)
sport(softbal)
sport(golf)
sport(basketball)
sport(ski)
sport(cricket)
sport(bridg)
sport(card)
sport(hike)
sport(bike)
sport(dive)
sport(diver)
sport(camp)
race(itali)
race(polish)
hobby(travel)
hobby(movi)
hobby(game)
hobby(art)
hobby(place)
hobby(cook)
hobby(favorit)
hobby(cat)
hobby(dog)
hobby(trip)
hobby(travel)
hobby(hobbi)
resume(resum)
race(america)
race(american)
entertainment(music)
entertainment(entertainment)
entertainment(amus)
entertainment(pictur)
entertainment(theartr)
entertainment(stuff)
entertainment(album)
entertainment(humor)
entertainment(photo)
relatives(wife)
relatives(friend)
relatives(father)
relatives(daughter)
relatives(famili)
study(advisor)
study(student)
relatives(son)

sport(footbal)
race(russian)
race(china)
race(singapor)
race(nederlands)
race(taiwan)
race(franc)
race(francais)
race(japan)
race(japanes)
race(korea)
race(germani)
race(italiano)
resume(cv)
personal(life)
personal(my)
personal(myself)
personal(welcom)
personal(favorit)
personal(bookmark)
personal(galleri)
subjectcode(cs)
subjectcode(cse)
subjectcode(ee)
relatives(mom)
relatives(girlfriend)
relatives(brother)
relatives(sister)
relatives(friendship)
relatives(dad)
relatives(dadi)
entertainment(fun)
entertainment(cool)
entertainment(movi)
weather(weather)

B 3 Faculty Homepage

academic_place(faculti)
academic_place(institut)
academic_place(univers)
academic_place(depart)
teach(cours)
teach(subject)
teach(student)
academic_interest(research)
academic_interest(paper)
academic_interest(public)
academic_interest(subject)
academic_job(lectur)
academic_job(teach)
academic_job(cours)
academic_activity(member)
academic_activity(acm)
academic_activity(ieee)
academic_activity(committee)
academic_activity(confer)

B 4 Project Homepage

project_def(project)
project_def(mission)
project_def(objective)
project_def(propos)
project_group(people)
project_group(group)
project_group(member)
project_group(researcher)
project_group(people)
project_group(manager)
project_group(alumni)
project_place(laboratori)
project_place(lab)

B 5 Adverse Homepage

adverse(advers)	symtom(hematolo)	symtom(acut)
adverse(interac)	symtom(vascular)	symtom(liver)
adverse(reaction)	symtom(cardiovascular)	symtom(metabol)
symtom(sleepi)	symtom(digest)	symtom(hepat)
symtom(nervou)	symtom(allerg)	symtom(urinari)

B 6 Overdose Homepage

overdose(overdosag)	contraindicate(contraind)	contraindicate(hypertension)
overdose(contraind)	contraindicate(hypersensitiviti)	contraindicate(allergic)
effect(fatal)	contraindicate(peptic)	contraindicate(hypertrophy)
effect(toxic)	contraindicate(ulcer)	contraindicate(bleed)
effect(coma)	contraindicate(hypoclycemia)	contraindicate(precnancy)
effect(vomit)	contraindicate(heart)	contraindicate(hypotension)

B 7 Warning Homepage

warning(warn)	targetpeople(labor)
warning(precaution)	targetpeople(dilivery)
targetpeople(pregnancy)	targetpeople(maternal)
targetpeople(mother)	targetpeople(animal)
targetpeople(nurse)	
targetpeople(pediatric)	

B 8 Patient Information Homepage

patientinfo(patient)	physician(physician)	usage(room)
information(inform)	physician(doctor)	usage(shake)
information(product)	patientinfo(take)	usage(breath)
information(prescription)	usage(temperatur)	usage(instruction)

B 9 Clinical Pharmacology Homepage

pharmacology(pharmacologi)	clinical(clinic)	druganalysis(neg)
pharmacology(pharmacodynam)	druganalysis(gram)	dilution(dilution)
pharmacology(pharmacokinet)	druganalysis(posit)	dilution(techniqu)

APPENDIX C

PUBLICATIONS

The ongoing of our research and contributions of this thesis were reported as several published papers as follows.

C.1 National Conference

1. Soonthornphisaj, N., and Kijsirikul, B. 2000. Web page classification using Incremental Iterative Cross-Training. In Proc. the 4th National Computer Science and Engineering Conference, pp. 113-117.

C.2 International Conferences

1. Soonthornphisaj, N., and Kijsirikul, B. 2000. Iterative Cross-Training: An algorithm for learning from unlabeled Web pages. In Proc. the 1st Int. Conf. Intelligent Technologies, pp. 55-63.

2. Kijsirikul, B.; Sasipongpairoege, P.; Soonthornphisaj, N.; and Meknavin, S. 2000. Supervised and unsupervised learning algorithms for Thai Web page identification. In Proc. Pacific Rim Int. Conf. on Artificial Intelligence, Australia, pp. 690-700.

3. Soonthornphisaj, N., and Kijsirikul, B. 2001. The effects of different feature sets on Web page categorization. In Proc. the 3rd Int. Conf. Enterprise and Information System, Portugal, pp. 404-410. (**Best Paper**)

4. Soonthornphisaj, N., and Kijsirikul, B. 2001. An evaluation of Incremental Iterative Cross-Training approach on Web page classification. In Proc. the 2nd Int. Conf. Intelligence Technologies, pp. 260-266. (**Best Paper Award**)

C.3 Book Chapter

1. Soonthornphisaj, N., and Kijsirikul, B. 2002. The effects of different feature sets on Web page categorization. Enterprise Information System III, Kluwer press.

C.4 International Journal

1. Soonthornphisaj, N., and Kijsirikul, B. 2003. Iterative Cross-Training: an algorithm for Web page categorization. Intelligent Data Analysis, to appear.

BIOGRAPHY

Name	Nuanwan Soonthornphisaj
Sex	Female
Date of Birth	October 26, 1970
Marital Status	Married
Work	Lecturer in Department of Computer Science, Kasetsart University
Work address	Department of Computer Science, Faculty of Science, Kasetsart University, Phaholyothin Road, Bangkok, 10900 Thailand.

Education

1997	M.Sc. in Computer Science, Asian Institute of Technology
1992	B.Sc. in Computer Science, Thammasat University