รายงานวิจัยฉบับสมบูรณ์

โครงการ การทำเหมืองเว็บไทยโดยเทคนิคการเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัย

Thai Web Mining Using Machine Learning and Inductive Logic Programming

โดย บุญเสริม กิจศิริกุล

พฤศจิกายน 2546

สัญญาเลขที่ RSA/08/2544

รายงานวิจัยฉบับสมบูรณ์

โครงการ การทำเหมืองเว็บไทยโดยเทคนิคการเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัย

Thai Web Mining using Machine Learning and Inductive Logic Programming

บุญเสริม กิจศิริกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณสำนักงานกองทุนสนับสนุนการวิจัยที่ได้ให้ทุนตลอดระยะเวลา 3 ปี (1 ธค. 2543 – 30 พย. 2546) สำหรับโครงการ "การทำเหมืองเว็บไทยโดยเทคนิคการเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัย" และขอขอบคุณ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ได้ให้โอกาส สถานที่ ตลอดจนทรัพยากรต่างๆ ที่ใช้ในงานวิจัย ขอขอบคุณผู้ช่วยวิจัยและผู้มีส่วนร่วมในโครงการนี้ทุกท่าน

บุญเสริม กิจศิริกุล

พฤศจิกายน 2546

บทคัดย่อ

รหัสโครงการ: RSA/08/2544

ชื่อโครงการ: การทำเหมืองเว็บไทยโดยเทคนิคการเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัย

ชื่อนักวิจัย: นายบุญเสริม กิจศิริกุล

E-mail Address: boonserm.k@chula.ac.th

ระยะเวลาโครงการ: 1 ธค. 2543 – 30 พย. 2546

ปัจจุบันการเติบโตของอินเตอร์เน็ตเป็นไปอย่างรวดเร็วมาก มีเว็บเพจจำนวนหลายพันล้านเพจที่เข้าถึงได้บนอินเตอร์เน็ตและมีเว็บเพจหลายล้านเพจเกิดขึ้นใหม่ทุกวัน ผู้ใช้ข้อมูลบนอินเตอร์เน็ตต้องใช้เวลาและความพยายามอย่างมากในการค้นหาเอกสารที่ต้องการ ระบบค้นหาเว็บในปัจจุบันครอบคลุมเอกสารเพียงบางส่วนของเอกสารทั้งหมดบนอินเตอร์เน็ต ระบบค้นหาเว็บเหล่านี้มักคืนได้เอกสารที่ไม่ตรงกับความต้องการของผู้ใช้เพราะใช้การค้นหาตามคำสำคัญ ส่วนระบบไดเร็กทอรีโครงข่าย เช่น Yahoo! จัดโครงสร้างของเว็บเพจแยกตามหมวดหมู่ของเว็บเพจ ทำให้สามารถค้นคืนเอกสารได้ตรงกับความต้องการของผู้ใช้มากกว่า อย่างไรก็ดี ระบบไดเร็กทอรีโครงข่ายก็มีข้อจำกัดที่ปริมาณเว็บเพจที่ครอบคลุมจะน้อยมาก เนื่องจากต้องใช้แรงงานคนจำนวนมากในการแบ่งหมวดหมู่ของเอกสาร

ในงานวิจัย เรานำเสนอวิธีการเพื่อแก้ปัญหานี้โดยการจำแนกประเภทของเว็บเพจออกเป็นหมวดหมู่อย่างอัตโนมัติ วิธีการที่นำเสนอนี้ใช้เทคนิคของการเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัย หัวข้อสำคัญที่เน้นทำวิจัยคือ (1) การวิจัยพื้นฐานเพื่อเพิ่มประสิทธิภาพของการโปรแกรมตรรกะเชิงอุปนัย และ (2) การวิจัยเทคนิคการเรียนรู้ของเครื่องที่สามารถใช้ประโยชน์จากข้อมูลแบบไม่มีฉลาก

การโปรแกรมตรรกะเชิงอุปนัย (ไอแอลพี) สามารถนำมาประยุกต์ใช้กับการจำแนกเว็บเพจเป็นหมวดหมู่โดยอัตโนมัติได้ และมีจุดเด่นอยู่ที่ผู้สอนสามารถป้อนความรู้เบื้องต้นในรูปแบบของโปรแกรมตรรกะอันดับที่หนึ่งได้ ซึ่งจะช่วยให้การจำแนกข้อมูลทำได้อย่างมีประสิทธิภาพยิ่งขึ้น ไอแอลพีจะให้เอ้าท์พุตเป็นเซตของกฎที่สอดคล้องกับตัวอย่างสอน อย่างไรก็ดีไอแอลพีมีข้อด้อยที่กฎที่สร้างได้อาจไม่ตรงพอดีกับตัวอย่างทดสอบ โดยเฉพาะข้อมูลที่มีสัญญาณรบกวน และทำให้ข้อมูลเหล่านี้ไม่สามารถจำแนกหมวดหมู่ได้อย่างถูกต้อง ในกรณีเช่นนี้ เราจำเป็นต้องใช้วิธีการที่สามารถหากฎที่ตรงกับข้อมูลมากที่สุด ในงานวิจัยนี้ เราใช้กระบวนการดึงลักษณะสำคัญและวิธีการแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก เพื่อหากฎที่ตรงกับข้อมูลมากที่สุด ผลการทดลองที่ได้แสดงให้เห็นว่า วิธีการดึงลักษณะสำคัญและแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กทำให้การโปรแกรมตรรกะเชิงอุปนัยมีประสิทธิภาพสูงขึ้น

นอกจากนั้นในงานวิจัยนี้เรายังได้นำเสนอวิธีการเรียนรู้แบบใหม่ที่เรียกว่า การสอนไขว้แบบวนซ้ำ ซึ่งสามารถใช้ประโยชน์จากข้อมูลไม่มีฉลากได้ วิธีการนี้มีข้อดีกว่าการเรียนรู้แบบสอนทั่วไปที่ต้องใช้ข้อมูลมีฉลากทั้งหมดและต้องอาศัยแรงงานคนจำนวนมากเพื่อติดฉลากให้กับข้อมูล แนวคิดของการสอนไขว้แบบวนซ้ำคือการรวมตัวแยกแยะย่อยสองตัว ซึ่งจะสอนโต้ตอบกันเองไปมาเพื่อปรับปรุงประสิทธิภาพของระบบโดยรวม เมื่อให้ข้อมูลไม่มีฉลากสองเซต แต่ละเซตสำหรับตัวแยกแยะย่อยแต่ละตัว ตัวแยกแยะจะติดฉลากให้กับตัวแยกแยะอีกตัว ด้วยการโต้ตอบที่ดีระหว่างตัวแยกแยะทั้งสอง ทำให้ประสิทธิภาพของระบบโดยรวมค่อยๆ ดีขึ้น ผลการทดลองแสดงให้เห็นว่าวิธีการที่นำเสนอสามารถใช้ประโยชน์จากข้อมูลที่ไม่มีฉลากได้อย่างมีประสิทธิภาพ เรายังได้ปรับปรุงประสิทธิภาพของการสอนไขว้แบบวนซ้ำโดยการใช้การโปรแกรมตรรกะเชิงอุปนัยมาเป็นตัวแยกแยะย่อย ผลที่ได้พบว่า อัลกอริทึมการสอนไขว้แบบวนซ้ำชนิดการโปรแกรมตรรกะเชิงอุปนัยสามารถเพิ่มประสิทธิภาพของการสอนไขว้แบบวนซ้ำแบบดั้งเดิมได้อย่างมาก และมีประสิทธิภาพสูงกว่าวิธีการอื่นๆ ทุกวิธีที่นำทดสอบรวมทั้งการเรียนรู้แบบสอนซึ่งใช้ข้อมูลแบบมีฉลากทั้งหมดอีกด้วย

คำหลัก: การทำเหมืองเว็บ, การโปรแกรมตรรกะเชิงอุปนัย, การเรียนรู้ของเครื่อง

Abstract

Project Code: RSA/08/2544

Project Title: Thai Web Mining using Machine Learning and Inductive Logic Programming

Investigator: Boonserm Kijsirikul

E-mail Address: boonserm.k@chula.ac.th

Project Period: December 1, 2000 – November 30, 2003

With the explosive growth of the Internet, today there are billions Web page accessible on the Internet with several million pages being added daily. The user must spend a great deal of time and effort looking for document he needs. Web search engines available today cover only some fraction of all documents in the Internet, and because of the use of keyword search, they also return documents not related to the real user-interest. Net directory systems, such as Yahoo!, organize their Web resources in category-specific style and thus can provide documents better matching the user needs. However, these systems have the limitations that the number of Web pages covered by the systems is even small as they need a lot of human effort to categorize the documents.

In this research, we propose an approach to solving this problem by automatically classifying Web pages into categories. The proposed approach employs the techniques of machine learning and inductive logic programming. Two main issues are studied in this research; (1) a basic research for improving techniques of inductive logic programming, and (2) machine learning techniques that can make use of unlabeled data.

Inductive Logic Programming (ILP) can be applied to automatic classification of Web pages. It has advantage that the user can provide background knowledge in the form of first-order logic programs. This makes more efficient classification of data. Given training examples and background knowledge as the input, ILP outputs a set of rules that is consistent with training examples. However, ILP has disadvantage that the obtained rules may not be exactly match with test data, especially noisy data, and thus the data cannot be correctly classified. In such a case, we need a method that finds the best matching rule. In this work, we employ first-order feature extraction and backpropagation neural networks to find the best matching rule. The experimental results show that feature extraction and the neural network improve the performance of ILP alone.

In the research, we also propose a new learning method, called Iterative Cross-Training (ICT) which can make use of unlabeled data. The method has advantage over traditional supervised learning which needs all labeled data and requires a lot of human effort to label the data. The idea of ICT is to combine two sub-classifiers which iteratively train each other for improving the performance of the whole system. Given two sets of unlabeled data, each of which is for each classifier, the classifier labels data for the other. With good interaction between two classifiers, the performance of the whole system is increasingly improved. The experimental results show that the proposed method can effectively use unlabeled data. We then further improve the performance of ICT by employing an ILP system as a sub-classifier. The results show that ICT with ILP gives significant improvement over the original ICT, and performs better than other methods tested in our experiment including the supervised learning method which uses all labeled data.

Keywords: Web Mining, Inductive Logic Programming, Machine Learning

II. เนื้อหางานวิจัย

ปัจจุบันการเติบโตของเว็บไซต์เป็นไปอย่างรวดเร็วมาก ในวันหนึ่งๆ มีเว็บเพจ (Web page) เกิดขึ้นใหม่ประมาณ 1.5 ล้านเพจ และมีเว็บเพจทั้งหมดมากกว่า 1,000 ล้านเพจ สำรวจในปี 2000 [Pierre, 2000] ทำให้ผู้ใช้ข้อมูลบนอินเตอร์เน็ตประสบความยากลำบากในการค้นหาข้อมูลที่ต้องการ ระบบค้นหาเว็บ (Web search engine) หนึ่งๆ ไม่สามารถจะค้นหาและเก็บข้อมูลได้ทั้งหมด เนื่องจากจำนวนเว็บเพจที่มีอยู่มากมายมหาศาล และการเพิ่มปริมาณที่รวดเร็วของเว็บเพจในแต่ละวัน สาเหตุหนึ่งที่ผู้ใช้มักไม่ได้ข้อมูลที่ต้องการก็เพราะระบบค้นหาส่วนใหญ่ เช่น Google[Google], Excite[Excite], AltaVista[AltaVista] ใช้การค้นหาตามคำสำคัญ (keyword) ทำให้เว็บเพจ (Web page) ที่ค้นคืนมาให้กับผู้ใช้มีเพจที่ไม่ตรงกับความต้องการของผู้ใช้จำนวนมาก แม้ว่าเว็บเพจนั้นจะมีคำสำคัญตรงกับที่ผู้ใช้ป้อนให้ก็ตาม Lawrence และ Giles [Lawrence & Giles, 1998] ได้ศึกษาพบว่า ระบบค้นหาเว็บที่มีชื่อเสียงเช่น Hotbot[Hotbot], AltaVista, Excite, infoseek[Infoseek] ค้นหาข้อมูลโดยได้การครอบคลุม (coverage) เพียงแค่ 57.5%, 46.5%, 23.1% และ 16.5% ตามลำดับ นอกจากอัตราการครอบคลุมที่ต่ำแล้ว ความแม่นยำ (precision) ก็น้อยด้วย กล่าวคือจะมีเพจที่ไม่เกี่ยวข้องติดออกมาจำนวนมาก

ตัวอย่างเช่น ผู้ใช้คนหนึ่งต้องการศึกษาว่าเสือ Jaguar มีน้ำหนักโดยเฉลี่ยเท่าไร และใช้คำสำคัญคือ "jaguar" และ "weight" เพื่อค้นหาข้อมูลบนอินเตอร์เน็ตจากระบบค้นหาเว็บ AltaVista ซึ่งมีเป็นระบบค้นหาเว็บที่มีการรวบรวมเว็บเพจไว้ทั้งสิ้นประมาณ 30 ล้านเพจ (สำรวจเมื่อปี 1996 [Ma, et al., 1996]) AltaVista จะให้เว็บเพจที่ค้นคืนมาได้ประมาณ 900 เอกสาร ซึ่งในจำนวนนี้มีเอกสารที่ไม่ตรงกับความต้องการจำนวนมาก เช่น มีข้อมูลของรถยนต์ยี่ห้อ Jaguar มีข้อมูลของทีมฟุตบอลที่ชื่อ Jacksonville Jaguars มีข้อมูลของเครื่องเล่นเกมส์ที่ชื่อ Atari Jaguar และอื่นๆ อีกมากมาย ทำให้ผู้ใช้ต้องสูญเสียเวลาจำนวนมากในการเลือกหาเอกสารที่ต้องการ ระบบค้นหาเว็บเหล่านี้แม้จะมีข้อเสียที่ให้เอกสารที่ไม่ตรงกับความต้องการจำนวนมาก แต่ก็มีข้อดีที่การสร้างระบบทำได้ค่อนข้างง่ายโดยไม่ต้องอาศัยแรงงานคนมากนัก ซึ่งจะใช้หุ่นยนต์เว็บ (Web robot) ที่เป็นโปรแกรมแบบหนึ่งทำหน้าที่รวบรวมเอกสารที่อยู่บนเครือข่ายอย่างอัตโนมัติ

นอกจากระบบค้นหาเว็บแล้ว ผู้ใช้ยังสามารถใช้ระบบไดเร็กทอรีโครงข่าย (net directory system) เพื่อค้นหาข้อมูลบนอินเตอร์เน็ตได้เช่นกัน ตัวอย่างที่มีชื่อเสียงของระบบประเภทนี้คือ Yahoo! ซึ่งสามารถค้นหาข้อมูลที่ตรงกับความต้องการได้มากกว่า เนื่องจากข้อมูลได้ถูกจัดแยกเป็นโครงสร้างตามหมวดหมู่ แต่ก็มีข้อเสียที่ปริมาณเว็บเพจที่ครอบคลุมจะน้อยกว่าระบบค้นหาเว็บมาก เนื่องจากต้องใช้แรงงานคนจำนวนมากในการจัดทำไดเร็กทอรีโครงข่าย (มีประมาณ 9 แสนเว็บเพจในการสำรวจเมื่อปี 1999 [Mladenic & Grobelnik, 1999])

ดังนั้นจะพบว่าปัญหาสำคัญ 2 ปัญหาที่เกิดขึ้นคือ (1) การครอบคลุมต่ำกล่าวคือระบบค้นหาเว็บหรือระบบไดเร็กทอรีโครงข่ายมีจำนวนเอกสารที่ค้นคืนมาได้ไม่ครอบคลุมเพียงพอ และ (2) ความแม่นยำน้อยกล่าวคือระบบค้นหาเว็บค้นคืนเอกสารที่ไม่ตรงกับความต้องการจำนวนมาก

งานวิจัยนี้มุ่งเน้นที่จะศึกษาวิจัยถึงแนวทางในการแก้ปัญหานี้ ซึ่งรวมข้อดีของระบบทั้งสองประเภทด้านบนโดยการจัดหมวดหมู่ของเว็บเพจให้ได้อย่างอัตโนมัติ เทคนิคที่จะนำมาใช้คือการเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัย และพัฒนาเทคนิคใหม่ของการเรียนรู้ของเครื่องที่สามารถเรียนรู้จากตัวอย่างที่ไม่มีฉลาก (unlabeled example) เนื่องจากพบว่าในโดเมนของเว็บเพจนั้น เรามีตัวอย่างที่ไม่มีฉลาก (ในกรณีนี้คือเว็บเพจ) จำนวนมากที่จะนำมาสอนระบบทำเหมืองเว็บ แต่ตัวอย่างที่มีฉลากนั้น ผู้สอนต้องทำการติดฉลากกับตัวอย่าง เช่น ถ้าต้องการสอนระบบทำเหมืองเว็บให้เรียนรู้ว่า อะไรคือเพจภาษาไทย อะไรคือเพจภาษาอื่น ผู้สอนต้องนำตัวอย่างที่ไม่มีฉลากมาติดฉลากลงไป ซึ่งเสียแรงงานจำนวนมาก และวิธีการเรียนรู้ของเครื่องที่จะ

นำมาใช้กับระบบทำเหมืองเว็บนั้น ส่วนมากต้องการตัวอย่างสอนจำนวนมากจึงจะเรียนรู้ได้อย่างถูกต้องแม่นยำ วิธีวิจัยที่จะทำในงานนี้ จึงมุ่งเน้นที่จะใช้ประโยชน์จากตัวอย่างที่ไม่มีฉลากให้มากที่สุด

งานวิจัยนี้นำเสนอโมเดลของการสอนแบบใหม่ที่เรียกว่า การสอนไขว้แบบวนซ้ำซึ่งใช้ประโยชน์จากตัวแยกแยะ (classifier) 2 ตัวคือ ตัวแยกแยะราคาแพง (expensive classifier) และตัวแยกแยะราคาถูก (cheap classifier) และจากชุดตัวอย่าง 2 ชุดคือ ชุดตัวอย่างที่ 1 และ ชุดตัวอย่างที่ 2 ซึ่งตัวอย่างทั้งสองชุดเป็นแบบไม่มีฉลากทั้งสิ้น ตัวแยกแยะทั้งสองจะมีพารามิเตอร์ที่ต้องเรียนจากตัวอย่าง ซึ่งถ้าเป็นตัวอย่างที่มีฉลากก็สามารถเรียนพารามิเตอร์ได้โดยตรงจากตัวอย่าง แต่ในกรณีที่เราสนใจคือตัวอย่างไม่มีฉลากนั้น ตัวแยกแยะทั้งสองก็น่าจะสามารถเรียนรู้พารามิเตอร์ได้เช่นเดียวกัน โดยอาศัยการโต้ตอบและการสอนระหว่างกันเอง เนื่องจากว่าเราต้องการใช้ประโยชน์จากตัวอย่างที่ไม่มีฉลาก ดังนั้นถ้าตัวแยกแยะไม่มีความรู้เบื้องต้นเกี่ยวกับโดเมนที่จะเรียนรู้เลย การเรียนรู้ก็ไม่สามารถทำได้ เราจึงให้ความรู้เบื้องต้นกับตัวแยกแยะด้วย ตัวอย่างเช่น ถ้าต้องการให้เรียนรู้ว่าเว็บเพจใดเป็นภาษาไทย เพจใดเป็นภาษาอื่น เราก็จะให้ความรู้เบื้องต้นกับตัวแยกแยะราคาแพงในรูปของพจนานุกรมภาษาไทย ตัวเรียนรู้ราคาแพงในที่นี้จึงหมายถึงตัวแยกแยะที่มีความรู้เกี่ยวกับโดเมนอยู่บ้าง และใช้การคำนวณมาก ส่วนตัวแยกแยะราคาถูกไม่จำเป็นต้องมีความรู้เบื้องต้นเกี่ยวกับโดเมน และใช้การคำนวณต่ำ ในงานวิจัยนี้ใช้ตัวแยกแยะแบบเบย์ (Bayes classifier) เป็นตัวแยกแยะราคาถูก ส่วนตัวแยกแยะราคาแพงใช้ตัวแยกแยะที่สามารถใช้ความรู้เบื้องต้นเกี่ยวกับโดเมนได้ เช่น อาจเป็นตัวแยกแยะตัดคำที่มีความรู้เบื้องต้นในรูปของพจนานุกรม หรือ อาจเป็นการโปรแกรมตรรกะเชิงอุปนัย – ไอแอลพี (Inductive Logic Programming – ILP) มีคุณสมบัติที่เหนือกว่าการเรียนรู้ของเครื่องชนิดอื่นที่สามารถรับความรู้เบื้องต้นจากผู้สอนได้ และจากการโต้ตอบและสอนกันเองระหว่างตัวแยกแยะทั้งสอง ซึ่งอาจต้องวนสอนกันเองหลายรอบจนกระทั่งพารามิเตอร์ของทั้งสองลู่เข้า การเรียนรู้ก็จะสิ้นสุด ข้อดีของตัวแยกแยะราคาถูกคือใช้การคำนวณต่ำดังนั้นเมื่อเรียนรู้สำเร็จแล้ว จึงเหมาะที่จะนำไปให้กับหุ่นยนต์เว็บใช้สำหรับงานที่ต้องการได้โดยไม่สิ้นเปลืองเวลาในการคัดเลือกเว็บเพจจากอินเตอร์เน็ต

วัตถุประสงค์

วัตถุประสงค์ของการวิจัยในโครงการนี้ได้แก่

(1) เพื่อทำการวิจัยพื้นฐานเพื่อเพิ่มประสิทธิภาพของการโปรแกรมตรรกะเชิงอุปนัย

(2) เพื่อวิจัยเทคนิคการเรียนรู้ของเครื่องที่สามารถใช้ประโยชน์จากข้อมูลแบบไม่มีฉลาก

(3) เพื่อปรับปรุงประสิทธิภาพของการสอนไขว้แบบวนซ้ำด้วยการโปรแกรมตรรกะเชิงอุปนัย

ระเบียบวิธีวิจัยและผลที่ได้

1. การวิจัยพื้นฐานเพื่อเพิ่มประสิทธิภาพของการโปรแกรมตรรกะเชิงอุปนัย

ในการจำแนกเว็บเพจออกเป็นหมวดหมู่โดยอัตโนมัตินั้น สามารถทำได้หลายวิธีการด้วยกัน วิธีการหนึ่งที่น่าสนใจคือ การโปรแกรมตรรกะเชิงอุปนัยหรือไอแอลพี (Inductive Logic Programming – ILP) เนื่องจากวิธีการนี้ ผู้สอนสามารถป้อนความรู้เบื้องต้นในรูปแบบของโปรแกรมตรรกะอันดับที่หนึ่งได้ ซึ่งจะช่วยให้การจำแนกข้อมูลทำได้อย่างมีประสิทธิภาพยิ่งขึ้น ในหัวข้อนี้จะกล่าวถึงการวิจัยพื้นฐานเพื่อเพิ่มประสิทธิภาพของไอแอลพี

การโปรแกรมตรรกะเชิงอุปนัย [Quinlan, 1990; Muggleton, 1991; Muggleton & De Raedt, 1994] โดยทั่วไปมักถูกนำไปใช้กับปัญหาที่มีลักษณะเป็นสองกลุ่ม (two-class concept) มีจุดหมายเพื่อจำแนกตัวอย่างออกเป็นสองกลุ่ม (class) คือ กลุ่มตัวอย่างบวก และกลุ่มตัวอย่างลบ โดยการสร้างกฎเพื่ออธิบายกลุ่มตัวอย่างบวก ดังนั้นเมื่อนำกฎที่สร้างได้ไปจำแนกตัวอย่างใหม่ ตัวอย่างที่ตรงกับกฎจะถูกจำแนกเป็นกลุ่มตัวอย่างบวก

ส่วนตัวอย่างที่ไม่ตรงกับกฎจะถูกจำแนกเป็นกลุ่มตัวอย่างลบ แต่ในกรณีที่ต้องการนำระบบไอแอลพีไปใช้จำแนกตัวอย่างออกเป็นหลายกลุ่ม (multi-class concept) นั้น อาจมีตัวอย่างบางตัวโดยเฉพาะตัวอย่างที่มีสัญญาณรบกวน (noisy data) ที่ไม่ตรงกับกฎข้อใดข้อหนึ่งในเซตของกฎทั้งหมด ซึ่งในกรณีเช่นนี้ระบบไอแอลพีแต่เพียงอย่างเดียวไม่สามารถจำแนกตัวอย่างในลักษณะนี้ได้ จึงจำเป็นต้องมีวิธีการอื่นเข้ามาช่วยเพื่อทำให้ระบบไอแอลพีสามารถจำแนกตัวอย่างออกเป็นหลายกลุ่มได้ ตัวอย่างของวิธีการเหล่านี้ได้แก่ วิธีการจำแนกตามกลุ่มหลัก (Majority class method) [Clark & Niblett, 1989;Dzeroski, et al., 1996] การใช้วิธีการของเบย์โดยระบบ 1BC [Flach & Lachiche, 1999] การสร้างต้นไม้ตัดสินใจที่สามารถใช้แทนกฎลำดับที่หนึ่ง (First-Order Rule) โดยระบบ TILDE [Blockeel & Raedt, 1997] ฯลฯ

อย่างไรก็ดียังไม่มีวิธีการใดที่สามารถเลือกกฎในกรณีที่ตัวอย่างนั้นไม่ตรงกับกฎข้อใดข้อหนึ่งได้ ดังนั้นในงานวิจัยนี้จึงเป็นการมุ่งศึกษาหาวิธีการ ซึ่งสามารถนำมาใช้จำแนกตัวอย่างเพื่อใช้ร่วมกับกฎที่ได้จากระบบไอแอลพี เราได้ใช้กระบวนการดึงลักษณะสำคัญและวิธีการแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก (Backpropagation Neural Network — BNN) เพื่อประมาณกฎสำหรับเลือกกลุ่มที่ใกล้เคียงในกรณีดังกล่าว กระบวนการดึงลักษณะสำคัญใช้เพื่อหาลักษณะสำคัญจากกฎลำดับที่หนึ่ง รูปแบบของปัญหาเดิมจะเปลี่ยนไปอยู่ในรูปของค่าคุณลักษณะ (attribute value) ซึ่งเป็นปัญหาที่มีลักษณะเป็นตรรกศาสตร์ประพจน์ (propositional logic) โดยใช้ค่าความจริงจากลักษณะสำคัญจัดเป็นอินพุตเวกเตอร์ (input vector) เพื่อป้อนให้แก่นิวรอล-เน็ตเวิร์กเรียนรู้และทดสอบ นิวรอลเน็ตเวิร์กเป็นวิธีการเรียนรู้ของเครื่องแบบหนึ่งซึ่งใช้กันอย่างแพร่หลาย เนื่องจากนิวรอลเน็ตเวิร์กสามารถเรียนรู้เพื่อปรับค่าน้ำหนักของเส้นเชื่อมภายในโครงสร้าง ซึ่งเป็นการกำหนดความสำคัญให้แก่เส้นเชื่อมต่างๆ ดังนั้นหากนำลักษณะสำคัญมาป้อนเป็นอินพุตให้แก่นิวรอลเน็ตเวิร์ก เมื่อผ่านกระบวนการเรียนรู้แล้ว นิวรอลเน็ตเวิร์กจะสามารถกำหนดได้ว่าลักษณะสำคัญข้อใดมีความสำคัญมากกว่าลักษณะสำคัญข้ออื่น สาเหตุสำคัญอีกประการหนึ่งที่เลือกใช้วิธีแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก คือ นิวรอล-เน็ตเวิร์กสามารถจำแนกตัวอย่างที่มีลักษณะเป็นหลายกลุ่มได้ ดังนั้นด้วยการนำกระบวนการดึงลักษณะสำคัญและแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กมาใช้ร่วมกับกฎจากระบบไอแอลพี จะทำให้เราสามารถนำกฎที่ได้จากระบบไอแอลพีมาจำแนกตัวอย่างในกรณีที่ตัวอย่างนั้นไม่ตรงกับกฎข้อใดข้อหนึ่งพอดีได้ อันเป็นการพัฒนาความสามารถของระบบไอแอลพีให้จัดการกับปัญหาที่มีลักษณะเป็นหลายกลุ่ม หรือใช้จำแนกตัวอย่างซึ่งไม่ตรงกับกฎข้อใดในเซตของกฎพอดีได้

การประมาณกฎจากระบบไอแอลพีด้วยวิธีการแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก แบ่งเป็น 3 ขั้นตอนหลัก คือ (1) การดึงลักษณะสำคัญ (Feature Extraction) (2) การสร้างโครงสร้างนิวรอลเน็ตเวิร์กจากกฎและลักษณะสำคัญ และ (3) การสอนนิวรอลเน็ตเวิร์ก ซึ่งมีวิธีการดังจะกล่าวต่อไปนี้

1.1 การดึงลักษณะสำคัญ

ปัญหาสำคัญของกฎลำดับที่หนึ่ง คือในกรณีปัญหาที่มีลักษณะเป็นหลายกลุ่ม เมื่อจำแนกตัวอย่างทดสอบจะพบว่าจะมีตัวอย่างบางตัวอย่างที่ไม่ตรงกับกฎข้อใดเลย ระบบไอแอลพีเพียงลำพังจะไม่สามารถบอกได้ว่าตัวอย่างนั้นควรจะถูกจำแนกเป็นกลุ่มใด ในการวิจัยครั้งนี้มีแนวคิดว่าเมื่อไม่มีกฎข้อใดที่ตรงกับตัวอย่างพอดี เราสามารถใช้การครอบคลุมเพียงบางส่วน (partially cover) จากกฎแต่ละข้อเมื่อนำกฎไปเทียบกับตัวอย่าง เพื่อจำแนกตัวอย่างออกเป็นกลุ่มได้ การครอบคลุมเพียงบางส่วนอาจครอบคลุมคุณสมบัติบางอย่างที่สำคัญของกฎในแต่ละข้อ ซึ่งนำไปสู่การจำแนกตัวอย่างนั้นๆ ได้ ในกรณีที่ไม่สามารถหากฎที่ตรงกับตัวอย่างพอดี กฎที่ควรจะตรงกับตัวอย่างที่สุดควรเป็นกฎที่มีลักษณะสำคัญตรงกับตัวอย่างนั้นมากกว่ากฎข้ออื่นๆ โดยส่วนที่ไม่ตรงกับกฎควรเป็นลักษณะที่ไม่สำคัญ ในการทดลองครั้งนี้ได้ใช้แบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กมาทำการเรียนรู้ เพื่อกำหนดน้ำหนักซึ่งเป็นการให้ความสำคัญกับลักษณะสำคัญแต่ละข้อของกฎ

ตัวอย่างเช่น กฎลำดับที่หนึ่งซึ่งทุกสัญพจน์ (literal) ภายในกฎข้อนั้นมีแต่ตัวแปรซึ่งอยู่ในส่วนหัวของกฎ เราสามารถดึงลักษณะสำคัญ โดยใช้แต่ละสัญพจน์ในกฎข้อนั้นเป็นลักษณะสำคัญของกฎได้ ตัวอย่างเช่น

```
mesh(A,1) :- not_important(A), not_loaded(A).
```

จากตัวอย่างซึ่งเป็นกฎลำดับที่หนึ่งข้างต้นจะเห็นว่า สัญพจน์แต่ละตัวมีแต่ตัวแปรซึ่งอยู่ในส่วนหัวทั้งสิ้น และไม่มีตัวแปรใหม่ปรากฏอยู่ในสัญพจน์นั้นเลย เราจึงสามารถใช้สัญพจน์แต่ละตัวเป็นลักษณะสำคัญได้ และเรียกสัญพจน์ที่มีลักษณะแบบนี้ว่า ลักษณะสำคัญเดี่ยว (singleton feature)

แต่โดยปกติแล้วกฎในลำดับที่หนึ่งที่ได้จากระบบไอแอลพีจะมีความซับซ้อนกว่าตัวอย่างข้างต้นมาก ดังนั้น จึงต้องหาวิธีการที่สามารถดึงลักษณะสำคัญที่ประกอบด้วยตัวแปรใหม่ในแต่ละสัญพจน์ให้ได้ ในระบบไอแอลพี โดยทั่วไปสัญพจน์ที่มีตัวแปรใหม่จะไม่สามารถอยู่โดดเดี่ยวโดยปราศจากสัญพจน์อื่นซึ่งสร้างตัวแปรนี้ขึ้นมา และสัญพจน์ส่วนใหญ่ที่สร้างตัวแปรใหม่ จะถูกรวมเข้าอยู่ในกฎลำดับที่หนึ่งเพื่อทำหน้าที่ส่งผ่านตัวแปรใหม่นั้น ไปยังสัญพจน์อื่นซึ่งทำหน้าที่ตรวจสอบคุณสมบัติเฉพาะของตัวอย่างนั้นๆ ดังนั้นในงานวิจัยนี้จึงได้นำลักษณะการเกิดตัวแปรใหม่ในสัญพจน์มาสร้างเป็นลำดับของสัญพจน์ เพื่อสร้างเป็นลักษณะสำคัญจากกฎลำดับที่หนึ่ง

เราได้คำนิยามสายสัญพจน์ปิด (closed chain) และสายสัญพจน์เปิด (open chain) เพื่อใช้ในกระบวนการดึงลักษณะสำคัญดังนี้

นิยามที่ 1 (สายสัญพจน์ปิด) *สัญพจน์ใดๆ จะเป็นส่วนหนึ่งของสายสัญพจน์ปิดถ้าตัวแปรใหม่ทุกตัวในสัญพจน์นั้นซึ่งไม่อยู่ในส่วนหัวของกฎปรากฏในสัญพจน์อย่างน้อยสองสัญพจน์ และปรากฏในสัญพจน์อย่างน้อยหนึ่งสัญพจน์ร่วมกับตัวแปรในส่วนหัวของกฎหรือตัวแปรซึ่งเคยมีอยู่แล้วในสายสัญพจน์นั้นๆ* □

กล่าวอีกนัยหนึ่งคือ สายสัญพจน์ปิดจะเริ่มจากตัวแปรซึ่งอยู่ในส่วนหัวของกฎ แล้วตัวแปรเหล่านี้จะไปสร้างตัวแปรใหม่ในสัญพจน์ซึ่งอยู่ในกฎ ตัวแปรใหม่ที่ถูกสร้างขึ้นไปสร้างตัวแปรใหม่ขึ้นอีก เป็นลำดับอย่างนี้ไปจนสิ้นสุดที่สัญพจน์ซึ่งไม่มีการสร้างตัวแปรใหม่อีก สัญพจน์ที่ไม่มีการสร้างตัวแปรใหม่นี้โดยส่วนใหญ่จะเป็นสัญพจน์ที่บอกลักษณะพิเศษของตัวอย่างนั้น ตัวอย่างเช่น จากกฎ

```
mesh(A,1) :- short(A), neighbour_yz_1(B,A), usual(B).
```

จะเห็นว่า ในส่วนหัวของกฎมีตัวแปร A ปรากฏอยู่ สัญพจน์ short(A) ซึ่งมีแต่ตัวแปรที่อยู่ในส่วนหัวเท่านั้น จะถูกจัดเป็นลักษณะสำคัญที่เป็นลักษณะสำคัญเดี่ยว นอกจากลักษณะสำคัญนี้แล้ว ลักษณะสำคัญอีกอันหนึ่งที่ถูกสร้างขึ้นในลักษณะของสายสัญพจน์ปิด คือ

```
neighbour_yz_1(B,A), usual(B).
```

ซึ่งเป็นลักษณะสำคัญที่เริ่มจากตัวแปร A ในส่วนหัวของกฎ จากนั้นที่สัญพจน์ neighbour_yz_1(B,A) ตัวแปร A สร้างตัวแปรใหม่ B ดังนั้นสัญพจน์ neighbour_yz_1(B,A) จะถูกรวมเข้าไว้ในสายสัญพจน์ จากนั้นตัวแปร B ซึ่งถูกสร้างขึ้นในสัญพจน์ neighbour_yz_1(B,A) ปรากฏอีกครั้งในสัญพจน์ usual(B) และในสัญพจน์ usual(B) นี้มีแต่ตัวแปรที่เคยเกิดขึ้นแล้ว และไม่มีการสร้างตัวแปรใหม่ ทำให้สายสัญพจน์นี้สิ้นสุดที่สัญพจน์ usual(B) และเป็นสายสัญพจน์ปิด จะเห็นได้ว่า จากตัวอย่างของสายสัญพจน์ปิดข้างต้น สัญพจน์ neighbour_yz_1(B,A) ปรากฏอยู่ในกฎข้อนี้เพื่อทำหน้าที่เพียงสร้างตัวแปร B ขึ้นจากตัวแปร A และส่งต่อตัวแปร B ไปยังสัญพจน์อื่น ในขณะที่สัญพจน์ usual(B) เป็นสัญพจน์ที่ใช้ระบุลักษณะเฉพาะของตัวอย่าง สัญพจน์ usual(B) จะปรากฏอยู่ในลักษณะสำคัญโดยไม่มีสัญพจน์ neighbour_yz_1(B,A) ร่วมอยู่ไม่ได้ ดังนั้นจึงต้องรวมทั้งสองสัญพจน์ไว้ในสายสัญพจน์ปิดด้วยกัน

นิยามที่ 2 (สายสัญพจน์เปิด) *สัญพจน์ใดๆ จะเป็นส่วนหนึ่งของสายสัญพจน์เปิด ถ้ามีตัวแปรใหม่ในสัญพจน์นั้นซึ่งไม่อยู่ในส่วนหัวของกฎ ปรากฏในสัญพจน์เพียงสัญพจน์เดียวเท่านั้นร่วมกับตัวแปรในส่วนหัวของกฎหรือตัวแปรซึ่งเคยมีอยู่แล้วในสายสัญพจน์นั้นๆ* □

ลักษณะการเกิดสายสัญพจน์เปิดในกฎลำดับที่หนึ่งจะคล้ายกับการเกิดสัญพจน์ปิด มีการนำลักษณะการเกิดตัวแปรใหม่มาใช้เช่นเดียวกันกับสายสัญพจน์ปิด โดยสัญพจน์ที่มีตัวแปรใหม่จะถูกรวมเข้าไว้ในสายสัญพจน์เช่นกัน เพียงแต่ในสายสัญพจน์เปิดจะมีตัวแปรใหม่เกิดขึ้นและตัวแปรใหม่นี้ไม่ปรากฏในสัญพจน์อื่น นั่นคือตัวแปรใหม่ที่เกิดขึ้นนี้ไม่ได้ทำหน้าที่สร้างตัวแปรอื่น หรือกำหนดลักษณะเฉพาะของตัวอย่าง ต่างจากในสายสัญพจน์ปิดซึ่งตัวแปรใหม่ทุกตัวที่เกิดขึ้นจะต้องทำหน้าที่อย่างใดอย่างหนึ่ง คือ สร้างตัวแปรอื่นหรือกำหนดลักษณะเฉพาะของตัวอย่าง ตัวอย่างของสายสัญพจน์เปิด เช่น

จากกฎ

```
mesh(A,2) :- short(A), neighbour_xy_1(B,A), neighbour_zx_1(C,B).
```

ลักษณะสำคัญหนึ่งที่ถูกสร้างขึ้นในลักษณะของสายสัญพจน์เปิด คือ

```
neighbour_xy_1(B,A), neighbour_zx_1(C,B).
```

จากตัวอย่างจะเห็นว่า ตัวแปร A ซึ่งอยู่ภายในส่วนหัวของกฎ ทำหน้าที่สร้างตัวแปรใหม่ B ขึ้นภายในสัญพจน์ neighbour_xy_1(B,A) และตัวแปร B ที่เกิดขึ้นใหม่ ทำหน้าที่สร้างตัวแปรใหม่ C ที่สัญพจน์ neighbour_zx_1(C,B) จะเห็นได้ว่าสัญพจน์ neighbour_xy_1(B,A) ทำหน้าที่ส่งต่อตัวแปร B เพื่อไปสร้างตัวแปรใหม่ C ยังสัญพจน์ถัดไป ดังนั้นสัญพจน์ neighbour_xy_1(B,A) จึงถูกรวมในสายสัญพจน์เปิด ส่วนในสัญพจน์ neighbour_zx_1(C,B) ตัวแปร C ที่ถูกสร้างขึ้นไม่ปรากฏอยู่ในสัญพจน์อื่นอีกเลย ดังนั้นสายสัญพจน์นี้จึงเป็นสายสัญพจน์เปิด

<u>ตัวอย่างสายสัญพจน์ปิดและสายสัญพจน์เปิด</u>

จากกฎ

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

จะได้สายสัญพจน์ปิดดังต่อไปนี้

```
1) q2(A,C),q3(C)
2) q2(A,C),q4(C,D),q5(D)
3) q2(A,C),q3(C),q4(C,D),q5(D)
4) q6(A,E,F),q9(E),q10(F,I),q11(I,B)
```

และจะได้สายสัญพจน์เปิดดังต่อไปนี้

```
1) q6(A,E,F),q7(E,G)
2) q6(A,E,F),q8(E,H)
3) q6(A,E,F),q7(E,G),q8(E,H)
4) q6(A,E,F),q7(E,G),q9(E)
5) q6(A,E,F),q8(E,H),q9(E)
6) q6(A,E,F),q7(E,G),q8(E,H),q9(E)
```

จากตัวอย่างจะเห็นว่า สัญพจน์บางตัวไม่สามารถปรากฏเพียงลำพังในสายสัญพจน์ได้ กล่าวคือ สัญพจน์ที่มีตัวแปรใหม่จะต้องปรากฏร่วมกับสัญพจน์อื่นที่มีตัวแปรซึ่งเคยปรากฏมาแล้ว หรือเป็นตัวแปรในส่วนหัวของกฎ หรือกล่าวอีกนัยหนึ่งว่า สัญพจน์ที่มีตัวแปรใหม่จะต้องปรากฏร่วมกับสัญพจน์ที่ทำหน้าที่สร้างตัวแปรนั้นซึ่งอาจเป็นสัญพจน์ในส่วนหัวของกฎหรือเป็นสัญพจน์ในตัวกฎก็ได้ เพราะถ้าสัญพจน์ซึ่งมีตัวแปรใหม่ปรากฏโดยไม่

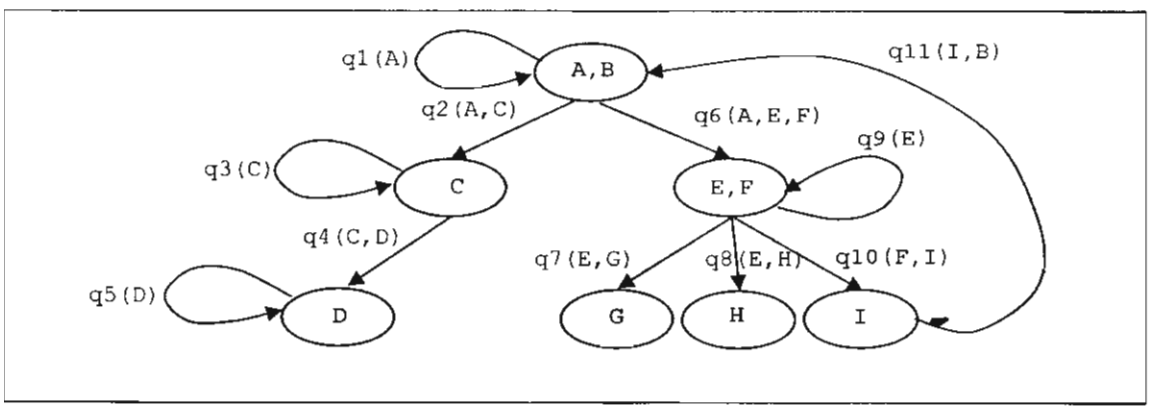
มีสัญพจน์ซึ่งทำหน้าที่สร้างตัวแปรนั้น จะทำให้ไม่สามารถหาการแทนที่ตัวแปรนั้นได้ ตัวอย่างของสายสัญพจน์เหล่านี้ เช่น "q3(C)", "q1(A), q3(C)", "q5(D)", "q1(A), q5(D)" ฯลฯ ตามคำนิยามของทั้งสายสัญพจน์ปิดและสายสัญพจน์เปิดจะจำกัดไม่ให้มีสายสัญพจน์เหล่านี้เกิดขึ้นอยู่แล้ว เนื่องจากจากคำนิยามของสายสัญพจน์ทั้งสองแบบ สัญพจน์ใดๆ จะถูกเข้ารวมเข้าไว้ในสายสัญพจน์ ถ้าในสัญพจน์นั้นมีตัวแปรใหม่ปรากฏร่วมกับตัวแปรที่มีอยู่แล้วในส่วนหัวของกฎหรือเคยปรากฏร่วมกับตัวแปรอื่นในสายสัญพจน์นั้นๆ

1.1.1 อัลกอริทึมการดึงลักษณะสำคัญ

จากคำนิยามของสายสัญพจน์ปิดและสายสัญพจน์เปิดจะเห็นได้ว่า สายสัญพจน์ทั้งสองมีแนวคิดมาจากการเกิดตัวแปรใหม่ภายในสัญพจน์ ดังนั้นเพื่อให้สามารถเข้าใจวิธีการดึงลักษณะสำคัญได้ง่าย ในงานวิจัยนี้จึงแทนกฎลำดับที่หนึ่งด้วยกราฟ โดยให้โนด (node) ของกราฟแทนตัวแปรใหม่ โนดเริ่มต้น (initial node) แทนตัวแปรซึ่งอยู่ในส่วนหัวของกฎ และเส้นเชื่อม (edge) แทนสัญพจน์ซึ่งมีตัวแปรนั้นๆ พิจารณากฎลำดับที่หนึ่งดังตัวอย่างต่อไปนี้

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

สัญพจน์ส่วนหัวคือ p(A,B) ดังนั้นโนดซึ่งแทนตัวแปร A และ B จะถูกจัดเป็นโนดเริ่มต้นของกราฟ สัญพจน์ q1(A) ไม่ได้ทำหน้าที่สร้างตัวแปรใหม่ มีเพียงตัวแปร A ปรากฏอยู่ภายในสัญพจน์ จึงมีเส้นเชื่อมซึ่งแทนสัญพจน์ q1(A) ออกและวนกลับเข้าที่โนดนั้น จากนั้นที่สัญพจน์ q2(A,C) ตัวแปร A ทำหน้าที่สร้างตัวแปรใหม่ C ดังนั้นจึงมีเส้นเชื่อมซึ่งแทนสัญพจน์ q2(A,C) ออกจากโนดเริ่มต้นไปยังโนดใหม่ซึ่งแทนตัวแปร C จากกฎข้างต้น เราสามารถแทนด้วยกราฟได้ดังรูปที่ 1



รูปที่ 1 ตัวอย่างของกราฟซึ่งใช้แทนกฎลำดับที่หนึ่ง

อัลกอริทึมการดึงลักษณะสำคัญมีขั้นตอนดังต่อไปนี้

(1) หาเส้นเชื่อมทุกเส้นซึ่งเริ่มต้นและสิ้นสุดที่โนดเริ่มต้น ใช้สัญพจน์เหล่านั้นเป็นลักษณะสำคัญเดี่ยว

(2) หาสายสัญพจน์ปิดทุกสายที่เป็นไปได้ซึ่งเริ่มต้นจากโนดเริ่มต้น สายสัญพจน์ที่ได้ในขั้นตอนนี้เป็นสายสัญพจน์ปิด

(3) หาทางเดินที่เป็นไปได้ทุกทางซึ่งเริ่มต้นจากโนดเริ่มต้นแล้วไปสิ้นสุดที่โนดสิ้นสุด (terminal node) ซึ่งเป็นโนดที่ไม่มีเส้นเชื่อมออกจากตัวเอง สายสัญพจน์ที่ได้ในขั้นตอนนี้เป็นสายสัญพจน์เปิด

(4) หาการรวม (combination) ทุกแบบที่เป็นไปได้ของสายสัญพจน์เปิดซึ่งได้จากขั้นที่สาม ที่มีตัวแปรใหม่ร่วมกัน

1.1.2 ตัวอย่างกระบวนการดึงลักษณะสำคัญ

พิจารณากฎด้านล่างนี้

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

จากกฎข้างต้นสามารถดึงลักษณะสำคัญได้ดังต่อไปนี้

ขั้นที่ 1

1) q1(A)

ขั้นที่ 2

2) q2(A,C),q3(C)

3) q2(A,C),q4(C,D),q5(D)

4) q2(A,C),q3(C),q4(C,D),q5(D)

5) q6(A,E,F),q9(E),q10(F,I),q11(I,B)

ขั้นที่ 3

6) q6(A,E,F),q7(E,G)

7) q6(A,E,F),q8(E,H)

8) q6(A,E,F),q9(E),q7(E,G)

9) q6(A,E,F),q9(E),q8(E,H)

ขั้นที่ 4

10) q6(A,E,F),q7(E,G),q8(E,H)

11) q6(A,E,F),q7(E,G),q8(E,H),q9(E)

จากตัวอย่างจะเห็นว่า อัลกอริทึมไม่ได้สร้างสายสัญพจน์เปิดทุกสายสัญพจน์ที่เป็นไปได้ โดยสายสัญพจน์เปิดที่ไม่ได้ถูกสร้างขึ้น ได้แก่ สายสัญพจน์เปิดที่เป็นส่วนหนึ่งของสายสัญพจน์ปิด เช่น "q2(A,C)", "q2(A,C),q4(C,D)" และ "q6(A,E,F),q9(E),q10(F,I)" ฯลฯ เนื่องจาก โดยปกติแล้วการสร้างตัวแปรใหม่ในกฎลำดับที่หนึ่ง มีจุดมุ่งหมายเพื่อใช้ตัวแปรใหม่นั้นในสัญพจน์อื่นซึ่งทำหน้าที่ตรวจสอบลักษณะเฉพาะของตัวอย่างนั้นๆ หรือทำหน้าที่สร้างตัวแปรใหม่ตัวอื่น ดังนั้นสายสัญพจน์ทุกสายสัญพจน์ที่สร้างขึ้นจึงควรสิ้นสุดที่สัญพจน์ซึ่งไม่มีตัวแปรใหม่เกิดขึ้น แต่ในบางกรณีที่สร้างสายสัญพจน์ปิดไม่ได้จึงจำเป็นต้องสร้างสายสัญพจน์เปิดด้วย

จากลักษณะสำคัญที่ได้ เราสามารถเปลี่ยนรูปแบบของกฎเพื่อนำไปใช้สร้างนิวรอลเน็ตเวิร์กโดยใช้ลักษณะสำคัญเดี่ยว สายสัญพจน์ปิด และสายสัญพจน์เปิด ประกอบเข้าไปในกฎเพื่อทำให้สามารถหาค่าความจริงของแต่ละลักษณะสำคัญได้โดยไม่ทำให้ความหมายเดิมของกฎเปลี่ยนไป ตัวอย่างเช่น จากตัวอย่างของกระบวนการดึงลักษณะสำคัญข้างต้น กฎที่ผ่านการเปลี่ยนรูปแบบแล้วจะเป็นดังรูปที่ 2

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

รูปที่ 2 ตัวอย่างของลักษณะสำคัญที่อยู่ภายในกฎ

1.2 การสร้างโครงสร้างนิวรอลเน็ตเวิร์กจากกฎและลักษณะสำคัญ

เมื่อผ่านกระบวนการดึงลักษณะสำคัญแล้ว ลักษณะสำคัญที่ได้จะอยู่ในรูปของลักษณะสำคัญเดี่ยว สายสัญพจน์ปิด และสายสัญพจน์เปิดดังรูปที่ 2 จากนั้นเราจะสร้างนิวรอลเน็ตเวิร์กโดยกำหนดให้ประกอบด้วย 3 ชั้น คือ ชั้นอินพุต (input layer) ชั้นซ่อน (hidden layer) และชั้นเอ้าท์พุต (output layer) นิวรอนในชั้นอินพุตแทนลักษณะสำคัญของกฎแต่ละข้อ ดังนั้นจำนวนนิวรอนที่มีในชั้นอินพุตจะเท่ากับจำนวนลักษณะสำคัญที่มีอยู่ในเซตของกฎ นิวรอนในชั้นซ่อนแทนกฎแต่ละข้อ จำนวนนิวรอนในชั้นนี้จึงมีค่าเท่ากับจำนวนกฎ นิวรอนในชั้นอินพุตซึ่งแทนลักษณะสำคัญภายในกฎแต่ละข้อจะมีเส้นเชื่อมมายังนิวรอนในชั้นซ่อนซึ่งแทนกฎข้อนั้นๆ นิวรอนในชั้นเอ้าท์พุตแทนกลุ่มของตัวอย่าง การเชื่อมต่อจากชั้นซ่อนมายังชั้นเอ้าท์พุตเป็นการเชื่อมต่อแบบทั้งหมด (fully connected) จำนวนนิวรอนในชั้นเอ้าท์พุตขึ้นกับจำนวนกลุ่มของตัวอย่าง ในกรณีของปัญหาแบบหลายกลุ่ม จำนวนนิวรอนในชั้นเอ้าท์พุตจะมีค่าเท่ากับจำนวนกลุ่ม ส่วนในกรณีของปัญหาแบบสองกลุ่ม จำนวนนิวรอนจะมีค่าเท่ากับสอง นิวรอนตัวหนึ่งจะแทนกลุ่มที่เป็นตัวอย่างบวก ในขณะที่นิวรอนอีกตัวหนึ่งแทนกลุ่มที่เป็นตัวอย่างลบ

<u>ตัวอย่างโครงสร้างของนิวรอลเน็ตเวิร์ก</u>

สมมติว่าในเซตของกฎประกอบด้วยกฎ 4 ข้อ คือ {*C1, C2, C3, C4*} และตัวอย่างแบ่งออกเป็น 3 กลุ่ม คือ {1, 2, 3} กฎข้อ *C1* เป็นกฎสำหรับรู้จำตัวอย่างกลุ่ม 1 กฎข้อ *C2* และ *C3* เป็นกฎสำหรับรู้จำตัวอย่างกลุ่ม 2 และ กฎข้อ *C4* เป็นกฎสำหรับรู้จำตัวอย่างกลุ่ม 3 ดังด้านล่างนี้

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

เมื่อผ่านกระบวนการดึงลักษณะสำคัญแล้วจะได้ลักษณะสำคัญดังต่อไปนี้

F1C1: `not_important(A)`
F2C1: `not_loaded(A)`

F1C2: `short(A)`
F2C2: `opposite_l(B,A)`

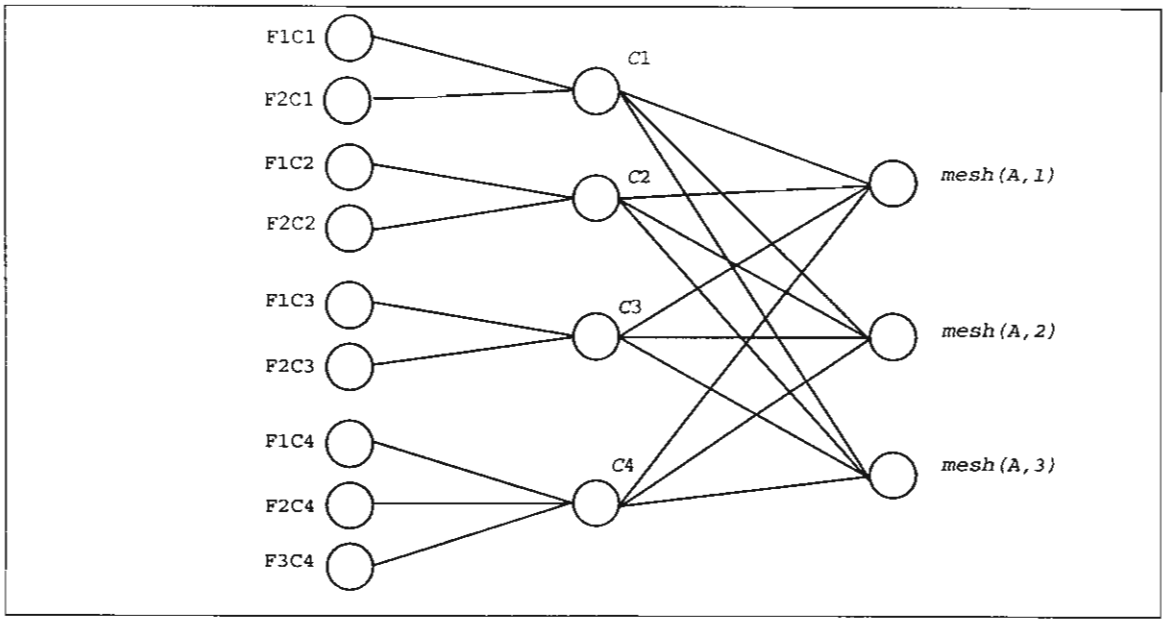
F1C3: `usual(A)`
F2C3: `neighbour_yz_r(A,B), cont_loaded(B)`

F1C4: `short(A)`
F2C4: `opposite_r(A,C), short(C)`
F3C4: `neighbour_zx_r(A,B)`

โดยที่ *FiCj* แทนลักษณะสำคัญที่ *i* ในกฎข้อที่ *j*

ในตัวอย่างด้านบนนี้ ลักษณะสำคัญ *F1C1, F2C1, F1C2, F1C3* และ *F1C4* เป็นลักษณะสำคัญเดี่ยว *F2C3* และ *F2C4* เป็นสายสัญพจน์ปิด ในขณะที่ *F2C2, F3C4* เป็นสายสัญพจน์เปิด ลักษณะสำคัญเดี่ยวเป็นการตรวจสอบคุณสมบัติของตัวอย่างโดยใช้เพียงสัญพจน์เดียว เช่น ในลักษณะสำคัญ *F2C1* "`not_loaded(A)`" เป็นการตรวจสอบว่า ตัวอย่าง `A` มีลักษณะเป็น `not_loaded(A)` หรือไม่ ตัวอย่างของสายสัญพจน์ปิด เช่น ลักษณะสำคัญ *F2C4* "`opposite_r(A,C), short(C)`" เป็นการตรวจสอบคุณสมบัติของตัวอย่าง `A` ว่ามีคุณสมบัติตรงตามสัญพจน์ทั้งสองหรือไม่ คือ `A` และ `C` มีความสัมพันธ์ `opposite_r(A,C)` กัน และ `C` ยังมีคุณสมบัติเป็น `short(C)` อีกด้วย จะเห็นในสายสัญพจน์นี้ได้ว่า ตัวแปร `C` ถูกสร้างขึ้นจากตัวแปร `A` ในสัญพจน์ `opposite_r(A,C)` ทำให้สัญพจน์ `short(C)` ไม่สามารถใช้เป็นลักษณะสำคัญโดยไม่มีสัญพจน์ที่สร้างมันขึ้นมา คือ `opposite_r(A,C)` ประกอบอยู่ด้วย ส่วนตัวอย่างของสายสัญพจน์เปิดในกรณีนี้คือ ลักษณะสำคัญ *F2C2* ซึ่งจำเป็นต้องใช้เป็นลักษณะสำคัญข้อหนึ่งด้วยเนื่องจาก `opposite_l(B,A)` เป็นการตรวจสอบคุณสมบัติว่าตัวอย่าง `A` มีคุณสมบัติ `opposite_l(B,A)` หรือไม่

ตัวอย่างนี้เป็นการจำแนกตัวอย่างออกเป็น 3 กลุ่ม ประกอบด้วย `mesh(A,1)`, `mesh(A,2)` และ `mesh(A,3)` ตามลำดับ ดังนั้นโครงสร้างของนิวรอลเน็ตเวิร์กจะเป็นตามรูปที่ 3 นั่นคือ เน็ตเวิร์กประกอบด้วย 3 ชั้น ชั้นอินพุต ชั้นซ่อน และชั้นเอ้าท์พุต ชั้นอินพุตประกอบด้วย 9 นิวรอน แต่ละนิวรอนแทนลักษณะสำคัญแต่ละข้อ (*F1C1, F2C1, F1C2, F2C2, F1C3, F2C3, F1C4, F2C4* และ *F3C4*) ชั้นซ่อนประกอบด้วย 4 นิวรอน แต่ละนิวรอนแทนกฎแต่ละข้อ (*C1, C2, C3* และ *C4*) ลักษณะสำคัญที่มาจากกฎข้อเดียวกันจะถูกเชื่อมต่อไปยังนิวรอนซึ่งแทนกฎข้อนั้นๆ เช่น นิวรอนซึ่งแทนลักษณะสำคัญ *F1C1* และ *F2C1* จะถูกเชื่อมต่อไปยังนิวรอนซึ่งแทนกฎ *C1* นิวรอนซึ่งแทนลักษณะสำคัญ *F1C2* และ *F2C2* จะถูกเชื่อมต่อไปยังนิวรอนซึ่งแทนกฎ *C2* เป็นต้น ส่วนที่ชั้นเอ้าท์พุต ประกอบด้วย 3 นิวรอน แต่ละนิวรอนแทนกลุ่มแต่ละกลุ่ม นิวรอนจากชั้นซ่อนและชั้นเอ้าท์พุตถูกเชื่อมต่อแบบทั้งหมด



รูปที่ 3 ตัวอย่างโครงสร้างของนิวรอลเน็ตเวิร์ก

1.3 การสอนนิวรอลเน็ตเวิร์ก

เมื่อกำหนดโครงสร้างของนิวรอลเน็ตเวิร์กได้แล้ว ขั้นตอนต่อไปคือ การสอนนิวรอลเน็ตเวิร์ก อินพุตเวกเตอร์ (input vector) ที่จะป้อนให้กับนิวรอลเน็ตเวิร์กได้มาจากการนำตัวอย่างที่ใช้สอนมาเทียบกับลักษณะสำคัญ เพื่อหาค่าความจริงของลักษณะสำคัญแต่ละข้อ โดยทั่วไปแล้วค่าความจริงของลักษณะสำคัญสามารถมีได้หลายแบบขึ้นกับการแทนที่ตัวแปรด้วยค่าคงที่ ในการวิจัยครั้งนี้ได้เลือกการแทนที่ที่ทำให้มีจำนวนลักษณะสำคัญที่มีค่าความจริงเป็นจริงมากที่สุด โดยกำหนดให้ค่าของอินพุตนิวรอนสำหรับลักษณะสำคัญที่มีค่าความจริงเป็นจริงเป็น 1 และลักษณะสำคัญที่มีค่าความจริงเป็นเท็จมีค่าเป็น –1 สำหรับส่วนของเอ้าท์พุตเวกเตอร์กำหนดให้เอ้าท์พุตนิวรอนที่แทนกลุ่มของตัวอย่างนั้นมีค่าเท่ากับ 1 นอกจากนั้นให้มีค่าเท่ากับ 0

<u>ตัวอย่างการสร้างอินพุตเวกเตอร์และเอ้าท์พุตเวกเตอร์</u>

จากตัวอย่างโครงสร้างของนิวรอลเน็ตเวิร์กข้างต้น กำหนดให้กลุ่มความรู้ภูมิหลังที่ใช้สร้างกฎเป็นดังนี้

```
not_important(a1).                not_loaded(a2).
short(a1).                        not_loaded(a3).
short(a2).                        cont_loaded(a2).
neighbour_yz_r(a1,a2).            cont_loaded(a3).
neighbour_zx_r(a1,a2).            opposite_r(a1,a2).
usual(a2).                        opposite_r(a1,a3).
```

เมื่อผ่านกระบวนการดึงลักษณะสำคัญในตัวอย่างข้างต้นแล้ว จะสามารถแปลงกฎแต่ละข้อเป็นดังนี้

```
C1:     mesh(A,1) :-
F1C1:           (not_important(A)),
F2C1:           (not_loaded(A)).

C2:     mesh(A,2) :-
F1C2:           (short(A)),
F2C2:           (opposite_l(B,A)).
```

```
C3:             mesh(A,2) :-
F1C3:           (usual(A)),
F2C3:           (neighbour_yz_r(A,B), cont_loaded(B)).

C4:       mesh(A,3) :-
F1C4:           (short(A)),
F2C4:           (neighbour_zx_r(A,B)),
F3C4:           (opposite_r(A,C), short(C)).
```

สมมติว่าใช้ตัวอย่าง mesh(a1,3) เป็นตัวอย่างที่ใช้เรียนรู้ เมื่อนำไปเทียบกับลักษณะสำคัญแต่ละข้อจะเป็นดังนี้

```
C1:       mesh(a1,1) :-
F1C1:         (not_important(a1)),                          TRUE
F2C1:         (not_loaded(a1)).                             FALSE

C2:       mesh(a1,2) :-
F1C2:         (short(a1)),                                  TRUE
F2C2:         (opposite_l(a2,a1)).                          FALSE

C3:       mesh(a1,2) :-
F1C3:         (usual(a1)),                                  FALSE
F2C3:         (neighbour_yz_r(a1,a2), cont_loaded(a2)).     TRUE

C4:       mesh(a1,3) :-
F1C4:         (short(a1)),                                  TRUE
F2C4:         (neighbour_zx_r(a1,a2)),                      TRUE
F3C4:         (opposite_r(a1,a2), short(a2)).               TRUE
```

เมื่อนำไปจัดเป็นอินพุตเวกเตอร์ให้กับนิวรอลเน็ตเวิร์กเพื่อทำการเรียนรู้ จะได้อินพุตเวกเตอร์เป็น <1, -1, 1, -1, -1, 1, 1, 1, 1> ส่วนของเอ้าท์พุตเวกเตอร์จะเป็น <0, 0, 1> เนื่องจากตัวอย่างนี้เป็นตัวอย่างของกลุ่ม 3 ซึ่งแทนด้วยเอ้าท์พุตนิวรอนตัวสุดท้าย ดังนั้นเอ้าท์พุตนิวรอนตัวสุดท้ายจึงมีค่าเป็น 1 นอกจากนั้นมีค่าเป็น 0 จะสังเกตจากตัวอย่างได้ว่า ในส่วนของลักษณะสำคัญจากกฎข้อ *C4* จะสามารถใช้การแทนที่ได้อีกแบบหนึ่งซึ่งให้ค่าความจริงไม่เหมือนกัน คือ

```
C4:       mesh(a1,3) :-
F1C4:         (short(a1)),                                  TRUE
F2C4:         (neighbour_zx_r(a1,a2)),                      TRUE
F3C4:         (opposite_r(a1,a3), short(a3)).               FALSE
```

ลักษณะสำคัญ *F3C4* สามารถใช้ค่าคงที่ a3 แทนตัวแปร C ได้ในสัญพจน์ opposite_r(A,C) ซึ่งทำให้ตัวแปร C ในสัญพจน์ short(C) ต้องถูกแทนที่ด้วยค่าคงที่ a3 ด้วยเช่นกัน ทำให้ลักษณะสำคัญ *F3C4* ที่ถูกแทนที่ด้วยค่าคงที่แล้วเปลี่ยนเป็น (opposite_r(a1,a3), short(a3)) ซึ่งให้ค่าความจริงเป็นเท็จ เมื่อเทียบกับการแทนที่โดยใช้ค่าคงที่ a2 แทนตัวแปร C ในตัวอย่างก่อนหน้านี้ที่ให้ค่าความจริงของลักษณะสำคัญในกฎข้อนี้เป็นจริงทั้งหมด ดังนั้นในตัวอย่างนี้จึงเลือกใช้การแทนที่ตัวแปร C ด้วยค่าคงที่ a2

ตัวอย่างที่ใช้ในกระบวนการเรียนรู้เพื่อสร้างกฎจากระบบไอแอลพี จะถูกนำมาเทียบกับลักษณะสำคัญเพื่อสร้างอินพุตเวกเตอร์และเอ้าท์พุตเวกเตอร์ดังตัวอย่างข้างต้น แต่ในระบบไอแอลพีโดยทั่วไปกฎที่ถูกสร้างขึ้นจะไม่ครอบคลุมตัวอย่างที่ใช้เรียนรู้ทุกตัว จะมีตัวอย่างบางตัวที่ไม่ตรงพอดีกับกฎข้อใดเลย ดังนั้นในการเลือกตัวอย่างที่ใช้เรียนรู้สำหรับนิวรอลเน็ตเวิร์ก จะมีเงื่อนไขดังต่อไปนี้

1) สำหรับปัญหาที่มีลักษณะเป็นหลายกลุ่ม จะเลือกเฉพาะตัวอย่างที่ถูกครอบคลุมด้วยกฎ หรือตัวอย่างที่ตรงพอดีกับกฎมาเป็นตัวอย่างสำหรับนิวรอลเน็ตเวิร์ก

2) ในปัญหาที่มีลักษณะเป็นสองกลุ่ม ตัวอย่างที่ใช้สำหรับสร้างนิวรอลเน็ตเวิร์กจะเป็นตัวอย่างบวกที่ตรงพอดีกับกฎและกลุ่มตัวอย่างลบ

กระบวนการเรียนรู้ของนิวรอลเน็ตเวิร์กจะเริ่มตั้งแต่ทำการสุ่มน้ำหนัก (weight) ของเส้นเชื่อม และค่าไบแอส (bias) ของโนด จากนั้นทำการเรียนรู้โดยทำการปรับน้ำหนักแบบเพิ่มขึ้น (incremental) โดยปรับน้ำหนักเมื่อป้อนตัวอย่างทีละตัว ด้วยขั้นตอนวิธีแบ็กพรอพาเกชัน [Rumelhart, et al., 1986] และทดสอบการลู่เข้า โดยนับจำนวนตัวอย่างที่ถูกติดกันทั้งหมดให้มากกว่าค่าที่กำหนดไว้ในตอนเริ่มทำการทดลอง นิวรอล-เน็ตเวิร์กที่สร้างได้จะถูกนำไปใช้รู้จำตัวอย่างต่อไป

1.4 ผลการทดลอง

ในส่วนนี้กล่าวถึงระบบไอแอลพีที่นำมาทดสอบ ชุดข้อมูลที่ใช้ในการทดลอง ผลการทดลองเปรียบเทียบระหว่างวิธีการประมาณกฎด้วยวิธีการแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก (Backpropagation Artificial Neural Network for Approximating Rules: BANNAR) กับวิธีการอื่นๆ ดังต่อไปนี้

1.4.1 ระบบไอแอลพีที่ใช้ในการทดลอง

เราได้ทดลองเปรียบเทียบระบบ BANNAR ซึ่งเราพัฒนาขึ้นกับระบบไอแอลพีอื่นๆ คือ PROGOL, GOLEM, TILDE, 1BC, LINUS และ FOSSIL ซึ่งรายละเอียดของแต่ละระบบมีดังนี้

1. PROGOL

ระบบ PROGOL [Muggleton, 1995] เป็นระบบไอแอลพีที่มีประสิทธิภาพระบบหนึ่งที่ใช้กันอย่างแพร่หลาย รับอินพุตเป็นเซตของตัวอย่างบวก ตัวอย่างลบ และกลุ่มความรู้ภูมิหลัง ผู้ใช้จะต้องประกาศลักษณะการทำงานของสัญพจน์ (mode declaration) เพื่อให้ระบบรู้ว่าแต่ละอาร์กิวเมนต์ของสัญพจน์จะมีลักษณะเป็นอย่างไร โดยอาร์กิวเมนต์ของสัญพจน์สามารถเป็นได้ทั้งตัวแปรอินพุต ตัวแปรเอ๊าท์พุต และค่าคงที่ นอกจากหน้าที่ของอาร์กิวเมนต์ต่างๆ แล้ว การประกาศลักษณะการทำงานของสัญพจน์นี้ยังสามารถระบุได้ถึงจำนวนครั้งที่ต้องการให้สัญพจน์นี้ปรากฏในกฎที่สร้างได้จากระบบอีกด้วย การสร้างกฎของระบบ PROGOL จะเริ่มตั้งแต่สุ่มตัวอย่างสร้างอนุประโยคที่เฉพาะเจาะจงที่สุด (most-specific) สำหรับตัวอย่างนั้น แล้วทำการค้นหาด้วยวิธีการแบบ A* (A*-like search) [Nilsson, 1980] โดยมีจุดมุ่งหมายเพื่อให้ได้การบีบอัด (compression) สูงสุดในการค้นหานั้น ทำให้ระบบ PROGOL ใช้เวลาและเนื้อที่หน่วยความจำมากในการสร้างกฎเมื่อเทียบกับระบบอื่นๆ ที่ใช้ในการทดลองครั้งนี้

เนื่องจากระบบ PROGOL เป็นระบบไอแอลพีที่ทำงานกับปัญหาที่มีลักษณะเป็นสองกลุ่ม คือ กลุ่มตัวอย่างบวก และกลุ่มตัวอย่างลบ ดังนั้นในการสร้างกฎสำหรับปัญหาที่มีลักษณะเป็นหลายกลุ่ม จึงสร้างกฎโดยสร้างครั้งละกลุ่ม กำหนดให้ตัวอย่างบวกในการเรียนรู้คือตัวอย่างในกลุ่มนั้น และตัวอย่างลบคือ ตัวอย่างของกลุ่มอื่นๆ ทำซ้ำจนครบทุกกลุ่ม จะได้กฎสำหรับทุกๆ กลุ่ม

2. GOLEM

ระบบ GOLEM [Muggleton & Feng, 1990] เป็นระบบไอแอลพีที่ทำงานกับปัญหาที่มีลักษณะเป็นสองกลุ่มเช่นเดียวกับระบบ PROGOL แต่แตกต่างกันในวิธีการค้นหาอนุประโยค โดยระบบ GOLEM รับอินพุตเป็นตัวอย่างบวก ตัวอย่างลบ และกลุ่มความรู้ภูมิหลัง ระบบเริ่มสร้างอนุประโยคด้วยการสุ่มเลือกตัวอย่างขึ้นมาเป็นคู่ หาอาร์แอลจีจีของตัวอย่างคู่นั้น เลือกผลการทำอาร์แอลจีจีที่ครอบคลุมตัวอย่างบวกมากที่สุด จากนั้นจึงวางนัยทั่วไปโดยนำอนุประโยคที่ได้มาทำอาร์แอลจีจีกับตัวอย่างบวกตัวใหม่ซึ่งสุ่มเลือกมาจากเซตของตัวอย่างบวกจนกระทั่งไม่สามารถหาอนุประโยคที่ครอบคลุมตัวอย่างได้มากขึ้น

ระบบ GOLEM อนุญาตให้ผู้ใช้ระบุจำนวนคู่ของตัวอย่างที่จะสุ่มมาทำอาร์แอลจีจีเพื่อหาอนุประโยคที่ครอบคลุมตัวอย่างมากที่สุดได้ ซึ่งจำนวนคู่ที่ระบบกำหนดไว้คือ 8 คู่ ถ้ากำหนดให้มีค่ามากขึ้น โอกาสที่ระบบจะพบอนุประโยคที่ครอบคลุมตัวอย่างได้มากจะสูงขึ้นตามไปด้วย ซึ่งในการทดลองครั้งนี้ได้ใช้ค่าที่ระบบกำหนดไว้ให้ คือ 8 คู่ในการสร้างกฎ และเนื่องจากระบบ GOLEM เป็นระบบที่ถูกสร้างขึ้นมาให้ใช้กับปัญหาที่เป็นสองกลุ่ม ดังนั้นในการสร้างกฎสำหรับกลุ่มใดๆ จะใช้ตัวอย่างของกลุ่มนั้นเป็นตัวอย่างบวกและใช้ตัวอย่างของกลุ่มอื่นๆ ที่เหลือเป็นตัวอย่างลบ เช่นเดียวกับขั้นตอนการสร้างกฎของระบบ PROGOL

3. TILDE

ระบบ TILDE [Blockeel & Raedt, 1997] จะทำการเรียนรู้เพื่อสร้างต้นไม้ตัดสินใจ (decision tree) ที่สามารถแทนกฎลำดับที่หนึ่งได้ หลักการทำงานของระบบจะคล้ายกับอัลกอริทึมการสร้างต้นไม้ตัดสินใจแบบทั่วไปที่ใช้กับค่าตรรกศาสตร์ประพจน์ โดยเริ่มจากสร้างโนดขึ้นมาทดสอบเพื่อเปรียบเทียบค่าฮิวริสติก (heuristic) ของแต่ละโนด ระบบจะเลือกโนดทดสอบที่มีค่าฮิวริสติกดีที่สุดในขณะนั้นมาสร้างเป็นโนดจริงในต้นไม้ จากนั้นจะนำตัวอย่างทั้งหมดมาทำการทดสอบกับโนดนี้เพื่อแบ่งตัวอย่างออกเป็นสองกลุ่ม คือ กลุ่มที่ให้ค่าความจริงเป็นจริงและให้ค่าความจริงเป็นเท็จเมื่อเทียบกับโนดปัจจุบัน จากนั้นจึงสร้างโนดทดสอบขึ้นมาใหม่สำหรับทดสอบกับตัวอย่างทั้งสองกลุ่มแล้วทำการทดสอบไปเรื่อยๆ จนกระทั่งมีตัวอย่างเพียงกลุ่มเดียวเท่านั้นที่ตรงกับโนดนั้น ระบบ TILDE นี้จะต่างจากระบบ PROGOL และ GOLEM ตรงที่ระบบ TILDE สามารถรู้จำตัวอย่างที่มีลักษณะเป็นหลายกลุ่มได้ เนื่องจากที่โนดใบ (leaf node) ของต้นไม้ตัดสินใจที่สร้างขึ้นจะแทนตัวอย่างแต่ละกลุ่ม เมื่อตัวอย่างไปตกอยู่ที่กลุ่มใด ก็จะถูกจำแนกเป็นกลุ่มนั้น

4. 1BC

1BC [Flach & Lachiche, 1999] เป็นตัวแยกแยะแบบเบย์ (Bayesian Classifier) สามารถใช้กับตรรกะลำดับที่หนึ่งได้ ระบบนี้ใช้แนวคิดจากวิธีการของเบย์กับตรรกศาสตร์ประพจน์ ซึ่งใช้ค่าทางสถิติของตัวอย่างมาจำแนกตัวอย่างใหม่ เมื่อนำแนวคิดของเบย์มาใช้กับตรรกะลำดับที่หนึ่งจึงต้องมีการปรับเปลี่ยนวิธีการรู้จำ ระบบ 1BC แบ่งลักษณะของสัญพจน์ออกเป็นสองกลุ่มด้วยกันคือ กลุ่มที่เป็นสัญพจน์โครงสร้าง และกลุ่มที่เป็นสัญพจน์ที่บอกลักษณะ ซึ่งสัญพจน์ทั้งสองกลุ่มนี้ผู้ใช้ต้องเป็นผู้กำหนดให้ในตอนเริ่มเรียนรู้ จากนั้นระบบจะสร้างลักษณะสำคัญจากสัญพจน์ทั้งสองกลุ่ม แล้วจึงเรียนรู้ด้วยอัลกอริทึมเบย์อย่างง่าย (naive Bayes algorithm) นอกจากลักษณะของสัญพจน์ที่ผู้ใช้เป็นผู้กำหนดแล้ว ระบบ 1BC ยังอนุญาตให้ผู้ใช้กำหนดลักษณะของลักษณะสำคัญที่สร้างขึ้นได้ โดยระบุจำนวนสัญพจน์และจำนวนตัวแปรที่สามารถปรากฏได้ในลักษณะสำคัญข้อหนึ่งๆ ซึ่งในการทดลองครั้งนี้ใช้ค่าปกติของระบบคืออนุญาตให้มีจำนวนสัญพจน์และจำนวนตัวแปรเท่ากับ 3 ทั้งสองค่า แต่ในการทดลองกับปัญหาการวิเคราะห์ไฟไนต์เอลิเมนต์เกิดปัญหาหน่วยความจำไม่พอ เนื่องจากมีสัญพจน์ที่มีค่าทั้งสองเป็น 3 อยู่เป็นจำนวนมาก จึงกำหนดให้ค่าทั้งสองมีค่าเป็น 2

5. LINUS

ระบบ LINUS [Lavrac & Dzeroski, 1994] เป็นระบบไอแอลพีที่สามารถใช้กับปัญหาที่มีลักษณะเป็นหลายกลุ่มและมีเป็นตรรกะลำดับที่หนึ่งได้ โดยระบบจะเปลี่ยนรูปแบบของปัญหาจากตรรกะลำดับที่หนึ่งให้เป็นค่าคุณสมบัติ (attribute value) การทำงานของระบบจะเริ่มโดยการสร้างลักษณะสำคัญจากตัวอย่างและความรู้ภูมิหลังซึ่งอยู่ในรูปตรรกะลำดับที่หนึ่ง ใช้ลักษณะสำคัญเหล่านั้นเป็นค่าคุณสมบัติของตัวอย่างแต่ละตัว จากนั้นจึงใช้ระบบที่สามารถรู้จำตัวอย่างจากค่าคุณสมบัติได้ เช่น ASSISTANT, CN2, NEWGEM และ C4.5 ฯลฯ ในการทดลองเปรียบเทียบครั้งนี้ได้สร้างลักษณะสำคัญโดยระบบ LINUS แล้วนำลักษณะสำคัญซึ่งอยู่ในรูปค่าคุณสมบัติไปทำการรู้จำด้วยระบบ C4.5 [Quinlan, 1993] เนื่องจากระบบ C4.5 สามารถจำแนกตัวอย่างซึ่งมีลักษณะเป็นหลายกลุ่มได้

1.4.2 ชุดข้อมูลที่ใช้ในการทดลอง

ชุดข้อมูลหรือกลุ่มของปัญหาที่ใช้ในการทดลองครั้งนี้มี 4 ชุด คือ การรู้จำภาพตัวพิมพ์อักษรไทย การวิเคราะห์ไฟไนต์เอลิเมนต์ การวิเคราะห์ความสามารถก่อกลายพันธุ์ และการวิเคราะห์ตำแหน่งตัวหมากรุก รายละเอียดของกลุ่มปัญหาทั้งหมดเป็นดังนี้

1. การรู้จำภาพตัวพิมพ์อักษรไทย (Thai Optical Character Recognition: TCR)

กลุ่มตัวอย่างในข้อมูลชุดนี้ประกอบด้วย พยัญชนะ สระ วรรณยุกต์ และตัวเลขไทย รวมทั้งหมดเป็นตัวพิมพ์อักษรไทย 77 ตัวอักษร ใช้ตัวอักษร 2 แบบ 7 ขนาด รวมเป็นชุดตัวอย่างที่ใช้เรียนรู้ทั้งสิ้น 1,078 ตัว ภาพตัวอย่างถูกสแกนด้วยความละเอียด 300 จุดต่อนิ้ว (dpi) จากนั้นนำมาผ่านกระบวนการกำจัดสัญญาณรบกวน กระบวนการทำภาพให้บางเพื่อหาเวกเตอร์พื้นฐานซึ่งประกอบกันเป็นภาพตัวอักษรนั้น นำเวกเตอร์พื้นฐาน บริเวณที่เป็นจุดร่วมของเส้น บริเวณที่เป็นรอยหยักขึ้นและรอยหยักลง จัดเป็นตัวอย่างที่ใช้เรียนรู้ กลุ่มความรู้ภูมิหลังที่ใช้ในการทดลองประกอบด้วยกฎซึ่งอยู่ในรูปกฎลำดับที่หนึ่งทั้งหมด 55 ข้อ กฎแต่ละข้อเป็นลักษณะของเส้นที่ใช้แบ่งแยกตัวอักษร เช่น `headzone(A,B)` คือ ตัวอักษรที่มีเวกเตอร์พื้นฐาน A มีส่วนหัวอยู่บริเวณ B หรือ `headprim(A,B)` หมายถึง ส่วนหัวของภาพตัวอักษรที่มีเวกเตอร์พื้นฐาน A มีเวกเตอร์พื้นฐาน B เป็นส่วนหัวของตัวอักษร ฯลฯ ซึ่งรายละเอียดของตัวอย่างที่ใช้เรียนรู้และความรู้ภูมิหลังปรากฏอยู่ใน [Kijsirikul & Sinthupinyo, 1999] กลุ่มตัวอย่างที่ใช้ทดสอบเป็นภาพตัวอักษรกลุ่มเดียวกับตัวอย่างที่ใช้เรียนรู้ ซึ่งได้รับการเพิ่มสัญญาณรบกวนโดยนำไปถ่ายเอกสารแบบเข้มและแบบจาง จากนั้นจึงนำไปผ่านกระบวนการเช่นเดียวกับตัวอย่างที่ใช้เรียนรู้ นำเวกเตอร์พื้นฐาน บริเวณที่เป็นจุดร่วมของเส้น บริเวณที่เป็นรอยหยักขึ้นและรอยหยักลง มาจัดเป็นตัวอย่างเพื่อใช้ทดสอบ

2. การวิเคราะห์ไฟไนต์เอลิเมนต์ (Finite Element Mesh Design: FEM)

จุดมุ่งหมายของปัญหา FEM [Dolsak & Muggleton, 1992] คือ การสร้างกฎเพื่อวิเคราะห์ไฟไนต์เอลิเมนต์ในโครงสร้าง โดยอาศัยกลุ่มความรู้ภูมิหลังเป็นลักษณะต่างๆ ของโครงสร้าง ประกอบด้วยลักษณะของเส้นเชื่อม เช่น `long`, `short` และ `usual` ฯลฯ เงื่อนไขขอบเขต เช่น `free` และ `one_side_fixed` ฯลฯ โลด เช่น `cont_loaded` และ `one_sideloaded` ฯลฯ กลุ่มตัวอย่างประกอบด้วยโครงสร้าง 5 แบบ แบ่งตัวอย่างออกเป็น 13 กลุ่ม แต่ละกลุ่มคือ จำนวนองค์ประกอบ (element) ที่เหมาะสมของโครงสร้างนั้น โดยตัวอย่างแต่ละตัวอย่างถูกจัดอยู่ในรูปแบบ `mesh(Edge, Number)` เมื่อ `Edge` คือ ชื่อโครงสร้าง และ `Number` คือ จำนวนองค์ประกอบภายในโครงสร้างนั้น รวมจำนวนตัวอย่างทั้งหมด 278 ตัวอย่าง

3. การวิเคราะห์ความสามารถก่อกลายพันธุ์ (Mutagenesis: MUTA)

จุดมุ่งหมายของปัญหา MUTA [Srinivasan, et al.,1996] คือ การสร้างกฎเพื่อวิเคราะห์ความสามารถก่อกลายพันธุ์ของโมเลกุล โดยอาศัยกลุ่มความรู้ภูมิหลังเป็นลักษณะต่างๆ ภายในโมเลกุลนั้น ประกอบด้วยลักษณะของอะตอมและลักษณะโครงสร้างของโมเลกุล เช่น `benzene`, `carbon_6_ring`, `carbon_5_aromatic_ring`, `hetero_aromatic_6_ring`, `hetero_aromatic_5_ring`, `ring_size_6`, `ring_size_5`, `nitro`, `methyl`, `anthracene`, `phenanthrene` และ `ball3` ฯลฯ กลุ่มตัวอย่างประกอบด้วยตัวอย่างที่เป็นข้อมูลของโมเลกุลทั้งหมด 188 โมเลกุล แบ่งเป็น 125 โมเลกุลอยู่ในกลุ่มตัวอย่างบวก และ 63 โมเลกุลอยู่ในกลุ่มตัวอย่างลบ

4. การวิเคราะห์ตำแหน่งตัวหมากรุกสากล (King-Rook-King Chess Endgame: KRK)

ในข้อมูลชุด KRK [Muggleton, et al., 1989] เป็นการวิเคราะห์ตำแหน่งของตัวหมากบนกระดานหมากรุกสากลที่ไม่สามารถเกิดขึ้นได้ในขณะที่ฝ่ายสีขาวเป็นผู้เดินและมีหมากเหลืออยู่บนกระดาน 3 ตัว คือ ขุนฝ่ายขาว เรือฝ่ายขาว และ ขุนฝ่ายดำ ตัวอย่างที่ใช้อยู่ในรูป `illegal(WKf, WKr, WRf, WRr, BKf, BKr)`

เมื่อ `WKf`, `WKr`, `WRf`, `WRr`, `BKf` และ `BKr` คือ แถว (file) ของขุนฝ่ายขาว หลัก (rank) ของขุนฝ่ายขาว แถวของเรือฝ่ายขาว หลักของเรือฝ่ายขาว แถวของขุนฝ่ายดำ และหลักของขุนฝ่ายดำ กลุ่มความรู้ภูมิหลังคือความสัมพันธ์ของตำแหน่งบนกระดาน ในชุดข้อมูลนี้ใช้ความสัมพันธ์ 2 แบบ คือ `adj(X,Y)` และ `lt(X,Y)` ซึ่งแสดงถึงตำแหน่ง `X` และ `Y` ที่อยู่ติดกัน และตำแหน่งที่มีค่าน้อยกว่ากัน ตามลำดับ จำนวนตัวอย่างในข้อมูลชุดนี้ประกอบด้วยตัวอย่างทั้งหมด 10,000 ตัวอย่าง แบ่งเป็นตัวอย่างบวก 3,361 ตัว และตัวอย่างลบ 6,639 ตัว

1.4.3 ผลการทดลองที่ได้

การทดลองกับชุดข้อมูล FEM, MUTA และ KRK ใช้การทดลองโดยแบ่งตัวอย่างออกเป็น 3 เซตย่อยแบบสุ่ม แล้วทำการทดลองโดยใช้เซตย่อยหนึ่งเซตเป็นเซตตัวอย่างที่ใช้ทดสอบและใช้เซตตัวอย่างที่เหลืออีกสองเซตเป็นเซตตัวอย่างที่ใช้เรียนรู้ จากนั้นวนทำซ้ำโดยให้เซตตัวอย่างทุกเซตเป็นตัวอย่างทดสอบหนึ่งครั้ง แล้วหาค่าเฉลี่ยของผลที่ได้ (3-fold cross-validation: 3CV) สำหรับชุดข้อมูล TCR ใช้การทดสอบโดยสร้างเซตตัวอย่างที่ใช้ทดสอบจากเซตตัวอย่างที่ใช้เรียนรู้ แต่เพิ่มสัญญาณรบกวนโดยผ่านการถ่ายเอกสารแบบเข้มและแบบจาง การทดลองเริ่มด้วยสร้างกฎจากตัวอย่างที่ใช้เรียนรู้ด้วยระบบ PROGOL หรือ GOLEM ตามเซตตัวอย่างที่ใช้เรียนรู้ โดยในชุดข้อมูล FEM และ KRK ใช้ระบบ GOLEM ในการสร้างกฎ ส่วนในชุดข้อมูล TCR และ MUTA ใช้ระบบ PROGOL ในการสร้างกฎ จากนั้นนำกฎที่ได้มาหาลักษณะสำคัญเพื่อนำไปเทียบกับตัวอย่างสำหรับเรียนรู้และทดสอบ แล้วสร้างเป็นอินพุตเวกเตอร์และเอ๊าท์พุตเวกเตอร์ นำอินพุตเวกเตอร์และเอ๊าท์พุตเวกเตอร์มาเรียนรู้และทดสอบ ผลการทดลองเปรียบเทียบระหว่างระบบ BANNAR กับระบบอื่นเป็นดังตารางที่ 1

ตารางที่ 1 ผลการทดลองเปรียบเทียบระหว่าง BANNAR กับระบบไอแอลพีอื่นๆ

ชุดข้อมูล	จำนวนตัวอย่างที่ใช้เรียนรู้	จำนวนตัวอย่างที่ใช้ทดสอบ	จำนวนกลุ่ม	BANNAR	PROGOL หรือ GOLEM	TILDE	1BC	LINUS
TCR	1,074	2,143	77	94.40	72.00^{3}	88.57^{3}	77.23^{3}	66.54^{3}
FEM	278	3CV	13	64.45	57.80^{2}	58.02^{2}	46.73^{2}	60.45^{1}
MUTA	188	3CV	2	83.58	82.01^{0}	68.94^{1}	77.72^{0}	74.41^{1}
KRK	10,000	3CV	2	99.93	99.89^{0}	69.80^{3}	87.12^{3}	99.89^{0}

หมายเหตุ: ใช้การทดสอบค่าทีแบบทางเดียว (one-tailed paired t-test) ตัวเลขยกแสดงระดับความเชื่อมั่น (confidence level) ตัวเลข 1, 2 และ 3 แสดงถึงระดับความเชื่อมั่นที่มากกว่า 90.0%, 99.0% และ 99.5% ตามลำดับ ส่วนตัวเลข 0 แสดงถึงระดับความเชื่อมั่นที่ต่ำกว่า 90%

เนื่องจากระบบ PROGOL และ GOLEM เป็นระบบไอแอลพีที่ใช้กับปัญหาที่มีลักษณะเป็นสองกลุ่ม ดังนั้นเมื่อนำไปใช้กับปัญหาที่มีลักษณะเป็นหลายกลุ่ม และตัวอย่างที่ใช้ทดสอบไม่ตรงกับกฎข้อใดในเซตของกฎ PROGOL และ GOLEM ไม่สามารถทำการจำแนกตัวอย่างนั้นได้ จึงต้องอาศัยวิธีการอื่นมาใช้จำแนกตัวอย่างเหล่านั้น ในการทดลองครั้งนี้ได้เลือกใช้วิธีการจำแนกตามกลุ่มหลักมาใช้ในกรณีดังกล่าว ผลการทดลองที่ปรากฏในตารางที่ 1 ในส่วนของ PROGOL และ GOLEM เป็นผลการทดลองที่ใช้วิธีการจำแนกตามกลุ่มหลักมาใช้ โดยในกรณีของปัญหาที่เป็นสองกลุ่ม คือ MUTA และ KRK ตัวอย่างที่ไม่ตรงกับกฎข้อใดเลย จะถูกจำแนกเป็นกลุ่มลบ ตัวอย่างที่ตรงกับกฎตั้งแต่หนึ่งข้อขึ้นไป จะถูกจำแนกว่าเป็นตัวอย่างบวก แต่ในกรณีปัญหาที่เป็นหลายกลุ่ม จะเลือกกลุ่มโดยใช้วิธีจำแนกตามกลุ่มหลัก กล่าวคือ เมื่อตัวอย่างที่ใช้ทดสอบไม่ตรงกับกฎข้อใดในเซตของกลุ่ม จะทำการจำแนกเป็นกลุ่มที่มีจำนวนมากที่สุดในกลุ่มตัวอย่างที่ใช้เรียนรู้ ในส่วนของระบบ LINUS

ที่นำมาใช้ทดลองเปรียบเทียบในการทดลองครั้งนี้ ใช้ระบบ LINUS สร้างลักษณะสำคัญของชุดตัวอย่าง จากนั้นนำลักษณะสำคัญที่ได้ซึ่งอยู่ในรูปตรรกศาสตร์ประพจน์ไปเรียนรู้และทดสอบด้วยระบบ C4.5

ผลการทดลองในตารางที่ 1 แสดงให้เห็นว่าเปอร์เซ็นต์ความถูกต้องของระบบ PROGOL หรือ GOLEM เมื่อนำไปใช้กับปัญหาที่มีลักษณะเป็นสองกลุ่ม คือ MUTA และ KRK ให้ผลการรู้จำสูงกว่า TILDE ในขณะที่เมื่อนำไปใช้กับปัญหาที่มีลักษณะเป็นหลายกลุ่ม คือ TCR และ FEM TILDE ซึ่งโดยปกติจะใช้งานกับปัญหาที่มีลักษณะเป็นหลายกลุ่มอยู่แล้ว ให้ผลการรู้จำสูงกว่า PROGOL หรือ GOLEM เมื่อเทียบผลการรู้จำของ PROGOL หรือ GOLEM กับ 1BC พบว่า PROGOL หรือ GOLEM ให้ผลการรู้จำสูงกว่า 1BC ในชุดข้อมูล FEM, MUTA และ KRK ให้ผลการรู้จำต่ำกว่า 1BC ในชุดข้อมูล TCR เทียบผลการรู้จำของ PROGOL หรือ GOLEM กับระบบ LINUS ปรากฏว่า PROGOL หรือ GOLEM ให้ผลการรู้จำสูงกว่า LINUS ในชุดข้อมูล TCR และ MUTA ให้ผลการรู้จำเท่ากันในชุดข้อมูล KRK และสูงกว่าในชุดข้อมูล TCR

จากผลการทดลองเมื่อเทียบเปอร์เซ็นต์ความถูกต้องของ PROGOL หรือ GOLEM กับระบบอื่น 3 ระบบ (TILDE, 1BC และ LINUS) ใน 4 ชุดข้อมูล รวมเปรียบเทียบ 12 การทดลอง พบว่า ในกลุ่มปัญหาที่มีลักษณะเป็นหลายกลุ่ม (TCR และ FEM) PROGOL หรือ GOLEM ให้ผลการรู้จำสูงกว่าระบบอื่น 2 การทดลองจาก 6 การทดลอง ต่ำกว่า 4 การทดลองจาก 6 การทดลอง และเมื่อเปรียบเทียบในกลุ่มปัญหาที่มีลักษณะเป็นสองกลุ่ม (MUTA และ KRK) PROGOL หรือ GOLEM ให้ผลการรู้จำสูงกว่าระบบอื่น 5 การทดลองจาก 6 การทดลอง และให้ผลการรู้จำเท่ากัน 1 การทดลองจาก 6 การทดลอง จะเห็นได้ว่า กฎที่ได้จากระบบ PROGOL หรือ GOLEM จะสามารถใช้ได้ดีในกลุ่มปัญหาที่มีลักษณะเป็นสองกลุ่ม และใช้ได้ไม่ดีในกลุ่มปัญหาที่มีลักษณะเป็นหลายกลุ่ม เมื่อนำวิธีการแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กมาใช้ในระบบ BANNAR ทำให้เปอร์เซ็นต์ความถูกต้องสูงขึ้น และเมื่อเทียบเปอร์เซ็นต์ความถูกต้องของ BANNAR กับระบบอื่น พบว่า BANNAR ให้เปอร์เซ็นต์ความถูกต้องสูงกว่าระบบอื่นทุกระบบในทุกชุดข้อมูล และให้เปอร์เซ็นต์ความถูกต้องสูงกว่าแบบมีนัยสำคัญทางสถิติเมื่อเทียบกับระบบ PROGOL หรือ GOLEM ใน 2 ชุดข้อมูล (TCR และ FEM) ให้เปอร์เซ็นต์ความถูกต้องสูงกว่าแบบมีนัยสำคัญทางสถิติเมื่อเทียบกับ TILDE ในทุกชุดข้อมูล ให้เปอร์เซ็นต์ความถูกต้องสูงกว่าแบบมีนัยสำคัญทางสถิติเมื่อเทียบกับ 1BC ใน 3 ชุดข้อมูล (TCR, FEM และ KRK) และให้เปอร์เซ็นต์ความถูกต้องสูงกว่าแบบมีนัยสำคัญทางสถิติเมื่อเทียบกับ LINUS ใน 3 ชุดข้อมูล (TCR, FEM และ MUTA)

1.5 สรุปผลการวิจัยพื้นฐานเพื่อเพิ่มประสิทธิภาพของการโปรแกรมตรรกะเชิงอุปนัย

ผลการวิจัยที่ได้แสดงให้เห็นว่าการนำวิธีการดึงลักษณะสำคัญมาใช้ร่วมกับแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กทำให้ประสิทธิภาพของการโปรแกรมตรรกะเชิงอุปนัยดีขึ้น สามารถจำแนกตัวอย่างที่ไม่ตรงพอดีกับกฎดั้งเดิมได้เป็นอย่างดี และผลการทดลองที่ได้แสดงให้เห็นถึงเปอร์เซ็นต์ความถูกต้องที่เพิ่มขึ้นของระบบ BANNAR เมื่อเทียบกับกฎเดิมซึ่งได้จากระบบ PROGOL หรือ GOLEM และเมื่อเทียบกับระบบอื่นอีก 4 ระบบ ใน 4 ชุดข้อมูล รวมเปรียบเทียบ 16 ครั้ง ผลปรากฏว่า ระบบ PROGOL หรือ GOLEM ให้เปอร์เซ็นต์ความถูกต้องต่ำกว่าระบบ TILDE และ LINUS ในชุดข้อมูลที่เป็นแบบหลายกลุ่ม และให้เปอร์เซ็นต์ความถูกต้องสูงกว่าระบบ TILDE และ LINUS ในชุดข้อมูลที่มีลักษณะเป็นสองกลุ่ม ผลการทดลองนี้แสดงให้เห็นว่ากฎที่ได้จากระบบ PROGOL และ GOLEM สามารถใช้ได้เป็นอย่างดีในปัญหาที่มีลักษณะเป็นสองกลุ่ม แต่ให้เปอร์เซ็นต์ความถูกต้องไม่ดีเมื่อใช้กับปัญหาที่มีลักษณะเป็นหลายกลุ่ม เมื่อใช้กระบวนการดึงลักษณะสำคัญและนิวรอลเน็ตเวิร์กในระบบ BANNAR ทำให้เปอร์เซ็นต์ความถูกต้องสูงขึ้นและสูงกว่าระบบอื่นในทุกชุดข้อมูล โดยให้ความถูกต้องสูงกว่าอย่างมีนัยสำคัญทางสถิติสูงกว่า 90.0% จำนวน 12 ครั้ง จากการเปรียบเทียบ 16 ครั้ง และให้เปอร์เซ็นต์ความถูกต้องสูงกว่าอย่างมีนัยสำคัญทางสถิติต่ำกว่า 90.0% จำนวน 4 ครั้ง จากการเปรียบเทียบ 16 ครั้ง

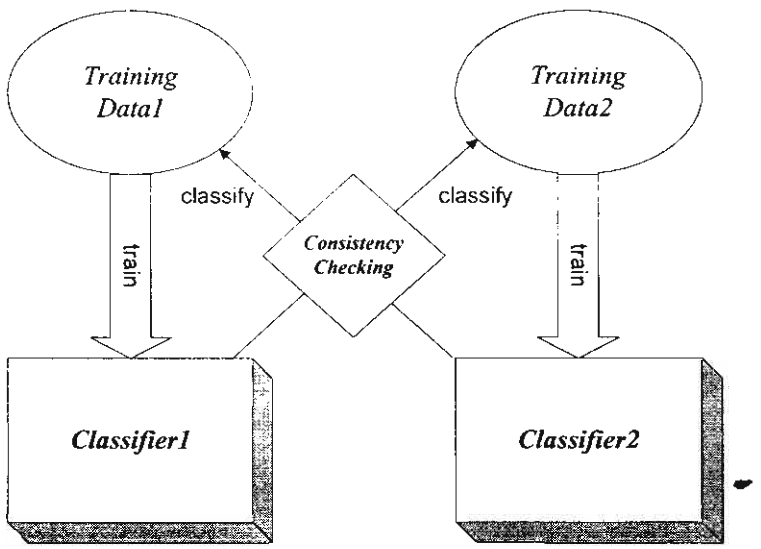
2. การวิจัยเทคนิคการเรียนรู้ของเครื่องที่สามารถใช้ประโยชน์จากข้อมูลแบบไม่มีฉลาก

วิธีที่นิยมใช้ในการจำแนกข้อมูลที่อยู่บนอินเตอร์เน็ตให้เป็นหมวดหมู่นั้น มักใช้วิธีการเรียนรู้โดยการสอน (supervised learning) ซึ่งสามารถทำได้โดยการติดฉลากเพื่อบอกหมวดหมู่ให้กับแต่ละเว็บเพจ และนำเว็บเพจเหล่านั้นไปสอนให้กับอัลกอริทึมในการเรียนรู้ แม้ว่าวิธีการเช่นนี้จะทำให้ได้ระบบจำแนกเว็บเพจที่มีความถูกต้องสูง แต่ก็จำเป็นต้องอาศัยแรงงานคนจำนวนมากเพื่อติดฉลากให้กับเว็บเพจ โดยเฉพาะอย่างยิ่งเว็บเพจบนอินเตอร์เน็ตมีการเปลี่ยนแปลงที่รวดเร็วมากทั้งในเชิงปริมาณที่เพิ่มขึ้นอย่างรวดเร็ว และในเชิงเนื้อหาที่มีการแก้ไขปรับปรุงกันอยู่เสมอ จึงเป็นเหตุให้การใช้แรงงานคนไม่สามารถทำได้อย่างมีประสิทธิภาพ

ในการวิจัยนี้เราต้องการทำให้ระบบค้นหาสามารถแยกหมวดหมู่ของเว็บเพจได้อย่างอัตโนมัติ โดยที่พยายามใช้ประโยชน์จากข้อมูลไม่มีฉลากให้มากที่สุด เพื่อลดแรงงานมนุษย์ เราจึงมุ่งเน้นที่จะหาวิธีการที่สามารถจำแนกประเภทของเว็บเพจอย่างอัตโนมัติ

งานวิจัยนี้นำเสนอเทคนิคใหม่ในการจำแนกเว็บเพจ วิธีการนี้มีชื่อว่า การสอนไขว้แบบวนซ้ำ (Iterative Cross-Training - ICT) วิธีการที่นำเสนอนี้สามารถใช้ประโยชน์จากข้อมูลที่ไม่มีฉลากได้อย่างมีประสิทธิภาพ เหมาะกับการใช้งานกับข้อมูลบนอินเตอร์เน็ตซึ่งมีเว็บเพจที่ไม่มีฉลากอยู่จำนวนมาก

2.1 อัลกอริทึมการสอนไขว้แบบวนซ้ำ



รูปที่ 4 การสอนไขว้แบบวนซ้ำ

รูปที่ 4 แสดงโมเดลของการสอนไขว้แบบวนซ้ำ ซึ่งประกอบด้วยตัวแยกแยะ (classifier) 2 ตัวคือ *Classifier1* และ *Classifier2* และชุดตัวอย่าง 2 ชุดคือ *TrainingData1* และ *TrainingData2* ซึ่งเป็นตัวอย่างที่ไม่มีฉลาก โดยการให้ความรู้เบื้องต้นเกี่ยวกับโดเมนที่จะเรียนรู้ หรือให้ตัวอย่างมีฉลากเริ่มต้นจำนวนน้อยๆ ตัวแยกแยะทั้งสองจะประมาณค่าของพารามิเตอร์ที่ต้องเรียนจากชุดตัวอย่าง โดยอาศัยการโต้ตอบและการสอนระหว่างกันเอง ตัวอย่างของความรู้เบื้องต้นก็อย่างเช่น ถ้าต้องการให้เรียนรู้ว่าเว็บเพจใดเป็นภาษาไทย เพจใดเป็นภาษาอื่น เราก็จะให้ความรู้เบื้องต้นในรูปของพจนานุกรมภาษาไทย ในกรณีที่เราไม่มีความรู้เบื้องต้นเกี่ยวกับโดเมนที่จะเรียนรู้ เราจะให้ตัวอย่างมีฉลากเริ่มต้นจำนวนน้อยๆ เพื่อให้โมเดลเริ่มกระบวนเรียนรู้ สำหรับชุดตัวอย่าง *TrainingData1* และ *TrainingData2* นั้นได้มาจากการทำสำเนาข้อมูลที่ผู้ใช้ให้มา ให้ θ_1 เป็นชุดพารามิเตอร์ (parameter set) ของ *Classifier1* และ θ_2 เป็นชุดพารามิเตอร์ของ *Classifier2* ชุดข้อมูล *TrainingData1* ใช้สำหรับสอน *Classifier1* ให้เรียนรู้พารามิเตอร์ ส่วน *TrainingData2* ใช้สำหรับสอน *Classifier2* อัลกอริทึมของการสอนไขว้แบบวนซ้ำแสดงในตารางที่ 2 ต่อไปนี้

ตารางที่ 2 อัลกอริทึมการสอนไขว้แบบวนซ้ำ

อินพุต:

- *TrainingData1* และ *TrainingData2* เป็นชุดของตัวอย่างไม่มีฉลาก

(1) กำหนดค่าเริ่มต้นของชุดพารามิเตอร์สำหรับ *Classifier1* เป็น θ_{10}

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

(2) กำหนดค่าเริ่มต้นของชุดพารามิเตอร์สำหรับ *Classifier2* เป็น θ_{20}

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

(3) วนซ้ำจนกระทั่ง θ_1 ไม่เปลี่ยนแปลงหรือจำนวนรอบเกินกว่าค่าขีดแบ่ง

- ใช้ *Classifier1* และชุดพารามิเตอร์ θ_1 เพื่อติดฉลากให้กับข้อมูลทุกตัวใน *TrainingData2* ให้เป็นตัวอย่างบวกและตัวอย่างลบ พร้อมกับตรวจเช็คความสอดคล้องของการแยกแยะกับ *Classifier2* ถ้าจำเป็น
- สอน *Classifier2* โดยใช้ตัวอย่างมีฉลากใน *TrainingData2* เพื่อประมาณค่าของชุดพารามิเตอร์ θ_2 ของ *Classifier2*
- ใช้ *Classifier2* และชุดพารามิเตอร์ θ_2 เพื่อติดฉลากให้กับข้อมูลทุกตัวใน *TrainingData1* ให้เป็นตัวอย่างบวกและตัวอย่างลบ พร้อมกับตรวจเช็คความสอดคล้องของการแยกแยะกับ *Classifier1* ถ้าจำเป็น
- สอน *Classifier1* โดยใช้ตัวอย่างมีฉลากใน *TrainingData1* เพื่อประมาณค่าของชุดพารามิเตอร์ θ_1 ของ *Classifier1*

แนวคิดของอัลกอริทึมด้านบนคือ หากเราสามารถดึงเอาข้อมูลทางสถิติที่น่าเชื่อถือออกมาจาก *TrainingData2* ได้ก็น่าจะมีประโยชน์ในการจำแนก *TrainingData1* ถ้าหากว่าชุดพารามิเตอร์เริ่มต้นของ *Classifier1*(θ_{10}) มีคุณสมบัติที่จะติดฉลากข้อมูลได้ถูกต้องมากกว่าติดฉลากผิดพลาดสำหรับ *TrainingData2* แล้ว เราก็น่าจะได้ข้อมูลทางสถิติที่แฝงอยู่ในตัวอย่างที่ทำนายถูก ซึ่งข้อมูลทางสถิตินี้เองที่จะช่วยให้ *Classifier2* สามารถติดฉลากได้อย่างถูกต้องให้กับตัวอย่างใน *TrainingData1* ที่มีคุณลักษณะคล้ายกัน และถ้า *TrainingData1* ที่มีการกำหนดฉลากใหม่นี้สามารถให้ θ_1 ที่ดีกว่า θ_{10} แล้ว เราก็น่าจะได้ชุดพารามิเตอร์ที่น่าเชื่อถือมากยิ่งๆ ขึ้นเมื่อผ่านไปในแต่ละรอบของการสอน ในขั้นตอนแรกของอัลกอริทึมนั้น เราต้องกำหนดค่าเริ่มต้นให้กับชุดพารามิเตอร์ของ *Classifier1* และ *Classifier2* ซึ่งทำได้โดยการสอนตัวแยกแยะด้วยข้อมูลจำนวนหนึ่งที่มีฉลาก (ถ้าหากเรามีข้อมูลเหล่านั้น) หรือถ้าเราไม่มีข้อมูลที่มีฉลากอยู่เลย ก็อาจกำหนดให้เป็นค่าใดๆ ที่มีการกำหนดไว้ล่วงหน้าหรืออาจกำหนดค่าโดยการสุ่มก็ได้

ในขณะที่ตัวแยกแยะที่กำลังทำงาน (active classifier) จะติดฉลากให้กับข้อมูล ตัวแยกแยะนั้นสามารถที่จะถามเพื่อยืนยันจากตัวแยกแยะอีกตัวได้ว่าตัวอย่างที่สนใจอยู่นั้นควรจัดอยู่ในกลุ่ม (class) ใด ถ้าหากตัวแยกแยะทั้งคู่เห็นพ้องกัน ตัวอย่างนั้นก็จะถูกกำหนดฉลาก แต่ถ้าหากว่าตัวแยกแยะทั้งคู่เห็นแตกต่างกันและตัวแยกแยะที่กำลังทำงานมีค่าความมั่นใจ (confident value) มากกว่า ตัวอย่างนั้นก็จะถูกกำหนดฉลากตามตัวแยกแยะที่กำลังทำงาน ถ้าไม่เช่นนั้นแล้วตัวอย่างนั้นก็จะไม่ถูกกำหนดฉลาก จุดมุ่งหมายของการตรวจเช็คความสอดคล้องกันนี้มีเพื่อให้การกำหนดฉลากของตัวอย่างมีความน่าเชื่อถือมากขึ้น อย่างไรก็ดีการตรวจเช็คนี้จะทำให้การทำงานของระบบช้าลง

เราได้นำวิธีการสอนไขว้แบบวนซ้ำประยุกต์ใช้กับปัญหาการจำแนกเว็บเพจ 2 ปัญหาคือ (1) การจำแนกประเภทเว็บเพจว่าเป็นเพจภาษาไทยหรือไม่ใช่ และ (2) การจำแนกประเภทเว็บเพจออกเป็นหมวดหมู่ ส่วน

ต่อไปจะกล่าวถึงรายละเอียดในปัญหาทั้งสองนี้ จากนั้นจะอธิบายถึงผลกระทบของข้อมูลสัญญาณรบกวนกับปัญหาการจำแนกประเภทเว็บเพจ

2.2 การจำแนกประเภทเว็บเพจภาษาไทยและภาษาอื่น

ในปัญหาของการจำแนกประเภทเว็บเพจภาษาไทยและไม่ใช่ภาษาไทยที่จะกล่าวในหัวข้อนี้ มีจุดมุ่งหมายคือเราต้องการจัดประเภทของเว็บเพจออกเป็นสองกลุ่ม คือ กลุ่มที่เป็นเว็บเพจภาษาไทยและกลุ่มที่ไม่ใช่ ซึ่งสามารถนำไปประยุกต์ใช้ในการสร้างหุ่นยนต์เว็บให้สามารถค้นหาเฉพาะเว็บเพจที่เป็นภาษาไทย เพื่อนำไปสร้างระบบค้นหาสำหรับเว็บเพจไทยได้ สำหรับปัญหานี้ตัวแยกแยะย่อยตัวแรกคือ *Classifier1* จะได้รับความรู้เบื้องต้นในรูปของพจนานุกรมไทย และใช้พจนานุกรมเพื่อช่วยในการจำแนกเว็บเพจว่าเป็นภาษาไทยหรือไม่ อัลกอริทึมที่ใช้ในตัวแยกแยะย่อยนี้คืออัลกอริทึมตัดคำภาษาไทย ส่วนตัวแยกแยะย่อยตัวที่สอง *Classifier2* จะไม่ได้รับความรู้เบื้องต้นและจะใช้อัลกอริทึมเบย์อย่างง่าย (Naïve Bayes) ดังจะกล่าวต่อไปนี้

2.2.1 ตัวแยกแยะในการจำแนกประเภทเว็บเพจภาษาไทยและภาษาอื่น

(1) ตัวแยกแยะตัดคำภาษาไทย

วิธีที่ง่ายที่สุดที่จะตรวจสอบว่าเว็บเพจเป็นเว็บเพจภาษาไทยหรือไม่ ทำได้โดยตรวจสอบคำในเว็บเพจว่าเป็นคำไทยหรือไม่ ถ้ามีคำจำนวนมากปรากฏในพจนานุกรม ก็แสดงว่าเว็บเพจนั้นน่าจะเป็นภาษาไทย อย่างไรก็ดีเราคงคาดหวังไม่ได้ว่าคำทุกคำในเว็บเพจนั้นต้องปรากฏในพจนานุกรมทั้งหมด เนื่องจากว่าในเว็บเพจไทยนั้นมักจะมีคำในภาษาอื่น เช่น ภาษาอังกฤษ ปะปนอยู่ด้วย นอกจากนั้นแล้วมักจะมีคำเฉพาะ เช่น ชื่อคน ชื่อสถานที่ต่างๆ ปรากฏอยู่ด้วย ซึ่งคำเฉพาะเหล่านี้มักจะไม่มีอยู่ในพจนานุกรม ดังนั้นจึงจำเป็นต้องกำหนดให้ได้ว่าควรจะมีคำไทยปรากฏมากน้อยเท่าไร จึงจะจัดว่าเป็นเว็บเพจไทย ปัญหานี้มีความยุ่งยากมากขึ้นโดยเฉพาะในภาษาไทย ซึ่งคำเขียนต่อเนื่องกันโดยไม่มีเครื่องหมายวรรคตอนคั่น ด้านล่างนี้อธิบายวิธีการตัดคำภาษาไทยที่ใช้ในงานวิจัยนี้

กำหนดให้เอกสาร d มีตัวอักษร n ตัว $(c_1, c_2, ..., c_n)$ ตัวแยกแยะตัดคำของเราจะสร้างการตัดคำทุกแบบที่เป็นไปได้และเลือกการตัดคำที่ดีที่สุดซึ่งให้ค่าฟังก์ชันด้านล่างนี้น้อยที่สุด

$$\text{argmin} \sum_{i=1}^{m} \text{cost}(w_i) \qquad (1)$$

โดยที่ $\text{cost}(w_i) = \eta_1$ ถ้า w_i เป็นคำที่ปรากฏในพจนานุกรม

$= \eta_2$ ถ้า w_i เป็นสายอักขระที่ไม่ปรากฏในพจนานุกรม

สำหรับการทดลองในหัวข้อที่ 2.2.3 นั้น เรากำหนดให้ η_1 และ η_2 มีค่าเท่ากับ 1 และ 2 ตามลำดับ

หลังจากที่ได้การตัดคำที่ดีที่สุดแล้ว เอกสารจะประกอบด้วย (1) คำที่ปรากฏในพจนานุกรม และ (2) สายอักขระที่ไม่ปรากฏในพจนานุกรม เว็บเพจไทยควรจะเป็นเพจที่มีคำไทยจำนวนมากและสายอักขระที่ไม่รู้จักจำนวนน้อย เราจึงนิยาม *WordRatio* ให้มีค่าเท่ากับ

$$\frac{\text{จำนวนตัวอักษรที่อยู่ในคำทั้งหมด}}{\text{จำนวนตัวอักษรทั้งหมดที่อยู่ในเอกสาร}} \qquad (2)$$

เมื่อให้เซตของตัวอย่างบวกและตัวอย่างลบ ตัวแยกแยะตัดคำจะหาค่าขีดแบ่งของ *WordRatio* ที่ทำให้ตัวอย่างบวกและลบซึ่งถูกจัดจำพวกอย่างถูกต้องมีจำนวนสูงสุด เอกสารใดที่มี *WordRatio* มากกว่าค่าขีดแบ่งจะถูกจัดจำพวกให้เป็นตัวอย่างบวก (เว็บเพจไทย) แต่เอกสารที่มี *WordRatio* น้อยกว่าค่าขีดแบ่งจะเป็นตัวอย่างลบ (ไม่ใช่เว็บเพจไทย) และเราใช้ค่าขีดแบ่งของ *WordRatio* เป็นพารามิเตอร์ของตัวแยกแยะตัดคำ

จากการใช้ค่าขีดแบ่งของ *WordRatio* เพียงตัวเดียวเป็นพารามิเตอร์ เราสามารถหา θ_{10} ซึ่งติดฉลากให้กับตัวอย่างได้ถูกต้องมากกว่าติดผิดพลาดได้ เว็บเพจไทยควรมีค่า *WordRatio* สูงและเว็บเพจที่ไม่ใช่ภาษาไทยควรให้ค่าต่ำ ถ้าจำนวนเว็บเพจไทยและไม่ใช่ไทยในเซตของตัวอย่างมีจำนวนเท่ากัน เราจะได้ว่าทุกค่าของค่าขีดแบ่ง *WordRatio* จะจำแนกตัวอย่างได้ถูกต้องมากกว่าจำแนกผิดพลาด (ยกเว้น $\theta_{10} = 0.0$ และ $\theta_{10} = 1.0$ ซึ่งให้จำนวนตัวอย่างที่จำแนกถูกและผิดเท่ากัน) ในกรณีที่เว็บเพจไทยมีจำนวนน้อยกว่าเว็บเพจที่ไม่ใช่ภาษาไทยนั้น θ_{10} ที่มีค่ามาก (เช่น 0.7, 0.8 หรือ 0.9) จะจำแนกตัวอย่างได้ถูกต้องมากกว่าจำแนกผิดพลาด θ_{10} ที่มีค่าน้อยจะเหมาะกับกรณีที่เว็บเพจไทยมีจำนวนมากกว่าเพจที่ไม่ใช่ไทย

เราสามารถปรับค่า θ_1 ใหม่ได้หลังจากที่ *Classifier2* ติดฉลากให้กับข้อมูลใน *TrainingData1* ดังนี้

ให้ *SP* เป็นค่าต่ำสุดของ *WordRatio* ที่ได้จากตัวอย่างบวกทั้งหมด และ *LN* เป็นค่าสูงสุดของ *WordRatio* ที่ได้จากตัวอย่างลบทั้งหมด ในกรณีที่ $SP \geq LN$ เราสามารถปรับค่าของ θ_1 ได้โดยใช้สมการด้านล่างนี้

$$\theta_1 = \frac{SP+LN}{2} \tag{3}$$

พิจารณากรณีที่ $SP<LN$ ให้ $V_1=SP$, $V_n=LN$ และ $V_2,\ldots,V_{n-1}$ เป็นค่าที่อยู่ระหว่าง V_1 กับ V_n ($V_1 \leq V_2 \leq \cdots \leq V_{n-1} \leq V_n$) เราสามารถปรับค่า θ_1 ใหม่ได้ดังนี้

$$\theta_1 = \frac{V_{i^*}+V_{i^*+1}}{2} \tag{4}$$

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

โดยที่ V_k เป็นค่า *WordRatio* ของตัวอย่างที่ถูกติดฉลากเป็นบวก และ V_j เป็นค่าของตัวอย่างที่ถูกติดฉลากเป็นลบ และ $V_1 \leq V_k \leq V_i$, $V_{i+1} \leq V_j \leq V_n$

ถ้า *SP* มากกว่าหรือเท่ากับ *LN*, θ_1 จะสามารถแยกตัวอย่างบวกออกจากตัวอย่างลบได้โดยสมบูรณ์ แต่ถ้า *SP* น้อยกว่า *LN*, θ_1 จะให้จำนวนตัวอย่างที่แยกผิดพลาดน้อยที่สุด

(2) <u>ตัวแยกแยะเบย์แบบง่าย</u>

สำหรับการแยกแยะข้อความนั้น ตัวแยกแยะเบย์อย่างง่ายเป็นวิธีการที่ได้รับความนิยมและเป็นวิธีที่มีประสิทธิภาพมากที่สุดวิธีหนึ่ง [Mitchell, 1997] วิธีนี้จะใช้ "*ถุงคำ (bag-of-words)*" เพื่อแทนข้อความ ในปัญหาการแยกเว็บเพจออกเป็นภาษาไทยและไม่ใช่นั้น เราจะใช้ "*ถุงตัวอักษร (bag-of-characters)*" แทน ซึ่งถุงตัวอักษรจะเหมาะกับปัญหานี้ เพื่อให้หุ่นยนต์เว็บสามารถค้นหาเว็บเพจไทยได้อย่างมีประสิทธิภาพ เพราะว่าไม่ต้องอาศัยการตัดคำ ดังนั้นการทำงานจะรวดเร็วมาก แม้ว่าการแทนข้อความด้วยถุงตัวอักษรจะง่ายกว่าการแทนด้วยถุงคำก็ตาม แต่การทดลองในหัวข้อต่อไปได้แสดงให้เห็นว่าวิธีการนี้ทำงานได้อย่างดี

กำหนดให้ $L = \{l_1, l_2, \ldots, l_m\}$ เป็นเซตของฉลากและ $d=(c_1, c_2, \ldots, c_n)$ เป็นเอกสารที่มี n ตัวอักษร ฉลากแสดงกลุ่มที่น่าจะเป็นที่สุด (l^*) สามารถประมาณค่าได้โดยใช้เทคนิคของเบย์อย่างง่าย ดังสมการด้านล่างนี้

$$\begin{aligned} l^* &= \underset{l_j}{\text{argmax}} \; Pr(l_j \mid c_1,\ldots,c_n) \\ &= \underset{l_j}{\text{argmax}} \; \frac{Pr(l_j)Pr(c_1,\ldots,c_n \mid l_j)}{Pr(c_1,\ldots,c_n)} \\ &= \underset{l_j}{\text{argmax}} \; Pr(l_j)Pr(c_1,\ldots,c_n \mid l_j) \end{aligned} \tag{5}$$

ในปัญหาของการจำแนกประเภทของเว็บเพจไทยหรือไม่ใช่ไทยนี้ L คือเซตของฉลากภาษาไทยและไม่ใช่ภาษาไทย ส่วน $d = (c_1, c_2, \ldots, c_n)$ โดยทั่วไปแล้วจะมีค่าที่เป็นไปได้ทั้งหมดมากมายมหาศาล ทำให้การคำนวณ $Pr(c_1, c_2, \ldots, c_n \mid l_j)$ ต้องใช้ข้อมูลจำนวนมหาศาลเพื่อให้ได้ความน่าจะเป็นที่น่าเชื่อถือได้ ดังนั้นเพื่อที่จะลดจำนวนของข้อมูลและปรับปรุงการประมาณค่าความน่าจะเป็นให้มีความน่าเชื่อถือมากยิ่งขึ้น เรานิยมใช้สมมติฐานของเบย์ดังนี้คือ (1) สมมติฐานเกี่ยวกับการไม่ขึ้นต่อกันอย่างมีเงื่อนไข (conditional independent assumption) กล่าวคือ การปรากฏของตัวอักษรตัวหนึ่งๆ จะไม่ขึ้นกับตัวอักษรอื่นๆ ทั้งหมดในเอกสารเมื่อรู้กลุ่มของตัวอย่าง และ (2) สมมติฐานที่ว่าตำแหน่งของตัวอักษรในเอกสารหรือเว็บเพจไม่มีความสำคัญ กล่าวคือ การพบตัวอักษร "ก" อยู่ที่ตำแหน่งแรกสุดของเอกสารมีค่าเหมือนกับการพบ "ก" ที่ท้ายเอกสาร แน่นอนว่าสมมติฐานเหล่านี้มักจะไม่เป็นจริงในทางปฏิบัติ แต่ผลการทดลองของตัวแยกแยะเบย์อย่างง่ายได้แสดงให้เห็นถึงประสิทธิภาพที่สูงของตัวแยกแยะนี้ในงานประเภทการแยกแยะข้อความ [Joachims, 1998; McCallum, et al., 1998; Yang & Pederson, 1997]

จากสมมติฐานด้านบนทำให้เราสามารถเขียนสมการด้านบนใหม่ดังต่อไปนี้

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

$$= \underset{l_j}{\operatorname{argmax}} \; Pr(l_j) \prod_{i=1}^{n} Pr(c_i \mid l_j) \qquad (6)$$

ค่าความน่าจะเป็น $Pr(l_j)$ และ $Pr(c_i \mid l_j)$ คือชุดพารามิเตอร์ θ_2 สำหรับตัวแยกแยะเบย์อย่างง่ายและประมาณค่าได้จากเซตของตัวอย่างสอน $Pr(l_j)$ ประมาณค่าได้จากอัตราส่วนระหว่างจำนวนตัวอย่างที่อยู่ในกลุ่ม l_j กับจำนวนตัวอย่างทั้งหมด ส่วน $Pr(c_i \mid l_j)$ หรือความน่าจะเป็นที่เราจะเห็นตัวอักษร c_i เมื่อรู้ว่ากลุ่มคือ l_j ประมาณค่าได้ดังต่อไปนี้

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

โดยที่ $N(c_i, l_j)$ คือจำนวนครั้งที่ตัวอักษร c_i ปรากฏในเซตของตัวอย่างสอนจากกลุ่ม l_j
$N(l_j)$ คือ จำนวนตัวอักษรทั้งหมดในเซตของตัวอย่างสอนจากกลุ่ม l_j และ
T คือจำนวนตัวอักษรที่แตกต่างกันทั้งหมดในเซตของตัวอย่างสอน

2.2.2 อัลกอริทึมที่นำมาเปรียบเทียบในการจำแนกประเภทเว็บเพจภาษาไทยและภาษาอื่น

ในการทดลองเพื่อประเมินประสิทธิภาพของการสอนไขว้แบบวนซ้ำนั้น เราได้เปรียบเทียบกับวิธีการต่อไปนี้

(1) ตัวแยกแยะตัดคำแบบสอน (Supervised Word Segmentation Classifier)
(2) ตัวแยกแยะเบย์อย่างง่ายแบบสอน (Supervised Naïve Bayes Classifier)
(3) อัลกอริทึมโคเทรนนิง (CoTraining Algorithm) และ
(4) อัลกอริทึมอีเอ็ม (EM Algorithm)

ตัวแยกแยะตัดคำแบบสอนและตัวแยกแยะเบย์อย่างง่ายแบบสอนมีอัลกอริทึมเหมือนกันกับตัวแยกแยะที่อธิบายไว้ในหัวข้อที่ 2.2.1 ยกเว้นว่าตัวแยกแยะในหัวข้อนี้จะใช้ข้อมูลที่มีฉลากทั้งหมดในการสอน ส่วนอัลกอริทึมโคเทรนนิงและอัลกอริทึมอีเอ็มเป็นดังต่อไปนี้

อัลกอริทึมโคเทรนนิง

อัลกอริทึมโคเทรนนิงถูกนำเสนอใน [Blum & Mitchell, 1998] แนวคิดของวิธีการนี้คือตัวอย่างสามารถมองได้สองมุมมอง เช่น เว็บเพจสามารถพิจารณาได้จากคำที่อยู่ในเนื้อหาของเพจนั้น หรือสามารถพิจารณาได้จากคำที่อยู่ในไฮเปอร์ลิงค์ของเพจที่ชี้มายังเพจนั้น และมีสมมติฐานที่ว่าแต่ละมุมมองเพียงพอต่อการเรียนรู้เพื่อจำแนกประเภทของตัวอย่าง อัลกอริทึมจะประกอบด้วยตัวแยกแยะย่อยสองตัว แต่ละตัวเรียนรู้จากคนละมุมมอง

จากแนวคิดนี้ เราได้ทดลองเขียนโปรแกรมตามอัลกอริทึมโคเทรนนิงดังแสดงในตารางที่ 3 ตัวแยกแยะย่อยในตารางมีอัลกอริทึมเหมือนกับของการสอนไขว้แบบวนซ้ำ ในการใช้อัลกอริทึมโคเทรนนิงในกรณีของปัญหาการจำแนกประเภทเว็บเพจภาษาไทยและไม่ใช่นั้น เรามองเว็บเพจแต่ละเพจเป็นเซตของคำที่ปรากฏในเพจนั้น หรือ เซตของตัวอักษรที่ปรากฏในเพจนั้น และใช้ตัวแยกแยะตัดคำ (*Classifier1*) เพื่อเรียนรู้จากเซตของคำ และใช้ตัวแยกแยะเบย์อย่างง่าย (*Classifier2*) เพื่อเรียนรู้จากเซตของตัวอักษรในเพจนั้น การปรับค่า θ_1 และ θ_2 ของตัวแยกแยะทั้งสองมีวิธีการเหมือนกับวิธีการที่อธิบายไว้ในหัวข้อที่ 2.2.1

ตารางที่ 3 อัลกอริทึมโคเทรนนิง

อินพุต:

- เซตของตัวอย่างที่มีฉลาก *LE* และ
- เซตของตัวอย่างที่ไม่มีฉลาก *UE*

(1) สร้างเซต *UE'* โดยเลือกตัวอย่าง *u* ตัวจากเซต *UE*

(2) วนซ้ำจนกระทั่งไม่มีตัวอย่างเหลืออยู่ใน *UE*

- ใช้ *UE* เพื่อประมาณค่าของชุดพารามิเตอร์ θ_1 ของ *Classifier1*
- ใช้ *UE* เพื่อประมาณค่าของชุดพารามิเตอร์ θ_2 ของ *Classifier2*
- ใช้ *Classifier1* พร้อมด้วยชุดพารามิเตอร์ θ_1 เพื่อติดฉลากให้กับตัวอย่างบวก *p* ตัวและตัวอย่างลบ *n* ตัวจาก *UE'*
- ใช้ *Classifier2* พร้อมด้วยชุดพารามิเตอร์ θ_2 เพื่อติดฉลากให้กับตัวอย่างบวก *p* ตัวและตัวอย่างลบ *n* ตัวจาก *UE'*
- นำตัวอย่างที่ติดฉลากใหม่นี้เพิ่มเข้าไปใน *LE*
- เลือกตัวอย่าง *2p+2n* ตัวอย่างสุ่มจาก *UE* และนำไปเพิ่มใน *UE'*

อัลกอริทึมอีเอ็ม

อัลกอริทึมอีเอ็ม (Expectation-Maximization – EM algorithm) ถูกนำเสนอใน [Dempster et al. 1977] อัลกอริทึมนี้เป็นอัลกอริทึมประเภทวนซ้ำแบบหนึ่ง เพื่อใช้แก้ปัญหาของข้อมูลไม่สมบูรณ์ (incomplete data) เมื่อกำหนดโมเดลของการสร้างข้อมูลและข้อมูลที่มีค่าบางส่วนหายไป อัลกอริทึมนี้จะใช้โมเดลปัจจุบัน เพื่อประมาณค่าที่หายไป แล้วใช้ค่าที่ประมาณได้เพื่อปรับโมเดลให้ดีขึ้น อัลกอริทึมจะคำนวณค่าความน่าจะเป็น (likelihood) ของพารามิเตอร์ของโมเดลที่ดีที่สุดแบบท้องถิ่น (local) เพื่อประมาณค่าที่หายไป ในกรณีที่เรานำอัลกอริทึมอีเอ็มมาใช้ในการเรียนรู้เว็บเพจที่ไม่มีฉลากนั้น ฉลากจะถูกมองว่าเป็นค่าที่หายไป อัลกอริทึมอีเอ็มแสดงในตารางที่ 4

ตารางที่ 4 อัลกอริทึมอีเอ็มสำหรับตัวแยกแยะเบย์อย่างง่าย

อินพุต:

- เซตของตัวอย่างที่ไม่มีฉลาก *UE*

(1) กำหนดค่าเริ่มต้นของชุดพารามิเตอร์ $Pr(c_i|l_j)$ และ $Pr(l_j)$ สำหรับ *Classifier* โดยเรียนจากตัวอย่างมีฉลากเริ่มต้น แล้วนำพารามิเตอร์เหล่านี้ไปติดฉลากให้กับตัวอย่างทั้งหมดใน *UE*

(2) ใช้ตัวอย่างที่ติดฉลากแล้ว ไปประมาณค่าพารามิเตอร์ $Pr(c_i|l_j)$ และ $Pr(l_j)$ ใหม่ของ *Classifier* โดยที่ $Pr(l_j|d) \in \{0,1\}$

(3) วนซ้ำจนกระทั่งพารามิเตอร์ไม่เปลี่ยนแปลงหรือจำนวนรอบเกินกว่าค่าขีดแบ่ง

- (E-step) ประมาณค่าฉลากของกลุ่มข้อมูลแบบถ่วงน้ำหนักตามความน่าจะเป็นให้เท่ากับ $Pr(l_j|d)$ สำหรับทุกเอกสาร ตามสมการที่ 10
- (M-step) ใช้ฉลากของกลุ่มข้อมูลที่ประมาณค่าได้ตาม $Pr(l_j|d)$ เพื่อคำนวณค่าพารามิเตอร์ใหม่โดยใช้เอกสารทั้งหมดตามสมการที่ 8 และ 9

ในงานวิจัยนี้โมเดลของการสร้างข้อมูลจะใช้ตัวแยกแยะเบย์อย่างง่าย อัลกอริทึมนี้ประกอบไปด้วย 2 ขั้นตอนคือ ขั้นตอนอี (E-step) และขั้นตอนเอ็ม (M-step) ขั้นตอนอีจะคำนวณฉลากกลุ่มข้อมูลแบบถ่วงน้ำหนักตามความน่าจะเป็น (probabilistically weighted class labels) สำหรับทุกเอกสารโดยใช้ตัวแยกแยะ และขั้นตอนเอ็มจะประมาณค่าพารามิเตอร์ใหม่โดยใช้เอกสารทั้งหมดที่ถูกคำนวณฉลากแบบถ่วงน้ำหนักแล้ว กระบวนการของขั้นตอนอีและขั้นตอนเอ็มจะวนซ้ำจนกระทั่งพารามิเตอร์มีค่าไม่เปลี่ยนแปลง Nigam และคณะ [Nigam et al. 1999] ได้ใช้อัลกอริทึมอีเอ็มสำหรับปัญหาการแยกแยะข้อความ

ดังที่แสดงในตารางที่ 4 ขั้นตอนแรกของอัลกอริทึมอีเอ็มคือ การประมาณค่าพารามิเตอร์ของตัวแยกแยะเบย์อย่างง่าย โดยเรียนจากตัวอย่างมีฉลากเริ่มต้น จากนั้นตัวแยกแยะจะใช้ค่าพารามิเตอร์ที่ได้นำไปติดฉลากให้กับตัวอย่างไม่มีฉลากทุกตัว หลังจากนั้นกระบวนการเรียนรู้จะวนทำขั้นตอนอี (E-step) และขั้นตอนเอ็ม (M-step) จนกระทั่งอัลกอริทึมลู่เข้า การประมาณค่าฉลากของกลุ่มข้อมูลแบบถ่วงน้ำหนักสามารถคำนวณได้ ดังต่อไปนี้

กำหนดให้ $L = \{l_1, l_2, ..., l_m\}$ เป็นเซตของฉลากและ $d = (c_1, c_2, ..., c_n)$ เป็นเอกสารที่มีตัวอักษร n ตัว จากชุดข้อมูลสอน D, $Pr(l_j|d) \in \{0, 1\}$ เป็นฉลากของเอกสาร d ค่าประมาณความน่าจะเป็นที่ตัวอักษร c_i จะอยู่ในฉลาก l_j คือ

$$Pr(c_i \mid l_j) = \frac{1 + \sum_{d \in D} N(c_i, d) Pr(l_j \mid d)}{T + \sum_{k=1}^{T} \sum_{d \in D} N(c_k, d) Pr(l_j \mid d)} \qquad (8)$$

โดยที่ T คือจำนวนตัวอักษรที่แตกต่างกันทั้งหมดในเซตของตัวอย่างสอน

$N(c_i, d)$ คือจำนวนครั้งที่ตัวอักษร c_i ปรากฏในเอกสาร d

ส่วนความน่าจะเป็นของฉลากหนึ่งๆ ที่จะเกิดขึ้นมีค่าตามสมการที่ (9) ต่อไปนี้

$$Pr(l_j) = \frac{1 + \sum_{d \in D} Pr(l_j \mid d)}{|L| + |D|} \quad (9)$$

โดยที่ $|L|$ และ $|D|$ คือจำนวนของฉลากที่แตกต่างกันและจำนวนเอกสารในชุดข้อมูลสอนตามลำดับ เมื่อกำหนดเอกสาร $d = (c_1, c_2, \ldots, c_n)$ ที่ไม่มีฉลากซึ่งมีตัวอักษร n ตัวให้ ตัวแยกแยะเบย์อย่างง่ายจะประมาณค่าความน่าจะเป็นที่เอกสารนี้มีฉลากเป็น l_j โดยใช้สมการ (10) ด้านล่างนี้

$$Pr(l_j \mid d) = \frac{Pr(l_j) Pr(d \mid l_j)}{Pr(d)} = \frac{Pr(l_j) \prod_{i=1}^{n} Pr(c_i \mid l_j)}{\sum_{k=1}^{|L|} Pr(l_k) \prod_{i=1}^{n} Pr(c_i \mid l_k)} \quad (10)$$

สังเกตว่าในสมการนี้ $Pr(l_j|d)$ เป็นค่าถ่วงน้ำหนักตามความน่าจะเป็น กล่าวคือแต่ละเอกสาร d จะถูกพิจารณาว่ามีฉลากเป็น l_j ด้วยความน่าจะเป็นเท่ากับ $Pr(l_j|d)$

2.2.3 ผลการทดลองการจำแนกประเภทเว็บเพจภาษาไทยและภาษาอื่น

ชุดข้อมูลและตั้งค่าการทดลองในปัญหาการแยกแยะเว็บเพจภาษาไทยและไม่ใช่

เรารวบรวมข้อมูลเพื่อทำการทดลองโดยเริ่มจากเว็บเพจ 4 เพจ คือ เว็บเพจภาษาญี่ปุ่น 1 เพจ[20] เว็บเพจภาษาไทย 2 เพจ[21] และ เว็บเพจภาษาอังกฤษ 1 เพจ[22] เริ่มต้นจากเว็บเพจทั้งสี่ เราให้หุ่นยนต์เว็บวิ่งตามไฮเปอร์ลิงค์ภายในเพจเหล่านั้น เพื่อเข้าถึงเว็บเพจอื่นๆ จนกระทั่งหุ่นยนต์เว็บรวบรวมเว็บเพจได้ 450 เพจสำหรับเว็บเพจเริ่มต้นแต่ละเพจ ดังนั้นเว็บเพจที่รวบรวมได้ทั้งสิ้นประกอบด้วยเว็บเพจภาษาไทย*ประมาณ* 900 เพจ เนื่องจากเพจภาษาไทยอาจจะชี้ไปยังเพจที่เป็นภาษาอังกฤษหรือภาษาอื่นๆ อีก ในทำนองเดียวกัน จะได้เว็บเพจที่เป็นภาษาญี่ปุ่นและภาษาอังกฤษอย่างละประมาณ 450 เพจ จากนั้นเรานำเว็บเพจทั้งหมดมารวมกันแล้วแบ่งออกเป็น 3 เซตคือ เซต A, B และ C แต่ละเซตมีเว็บเพจทั้งหมด 600 เพจ (เพจไทย อังกฤษและญี่ปุ่นประมาณ 300, 150 และ 150 ตามลำดับ) เราใช้วิธีการทดลองแบบ 3-fold cross-validation เพื่อวัดผลการทดลอง ในการทดลองแบบนี้ แต่ละเซตจะถูกนำมาเป็นชุดทดสอบเซตละหนึ่งครั้งโดยที่ใช้เซตที่เหลือเป็นชุดสอน แล้วหาค่าเฉลี่ยที่ได้เป็นผลการทดลอง

การตั้งค่าสำหรับอัลกอริทึมเป็นดังต่อไปนี้

- สำหรับการสอนไขว้แบบวนซ้ำนั้น ไม่มีการให้ตัวอย่างที่มีฉลากเริ่มต้นกับอัลกอริทึมการสอนไขว้แบบวนซ้ำ ค่าพารามิเตอร์เริ่มต้น θ, กำหนดให้มีค่าเป็น 0.7
- เนื่องจากอัลกอริทึมโคเทรนนิง (แสดงด้วย CoTraining ในตาราง) ต้องการตัวอย่างมีฉลากสำหรับเริ่มทำงาน ดังนั้นเราให้ตัวอย่างมีฉลากเริ่มต้นทั้งหมด 18 ตัว ในการทดลองพารามิเตอร์อื่นๆ ถูกกำหนดค่าดังต่อไปนี้ $|UE|$, p, n และ u มีค่าเป็น 1182, 3, 3 และ 115 ตามลำดับ
- สำหรับอัลกอริทึมอีเอ็ม กำหนดให้ค่าพารามิเตอร์เริ่มต้น θ, กำหนดให้มีค่าเป็น 0.7 จากนั้นใช้ตัวแยกแยะตัดคำ กำหนดฉลากให้กับตัวอย่างสอนของอัลกอริทึมอีเอ็ม หลังจากนั้นใช้อัลกอริทึมอีเอ็มด้วยตัวแยกแยะเบย์อย่างง่ายเพื่อเรียนรู้จนลู่เข้า

---

[20] http://www.yahoo.co.jp
[21] http://www.sanook.com, http://www.pantip.com
[22] http://www.javasoft.com

ผลที่ได้

ในการวัดประสิทธิภาพของวิธีการที่ทดสอบ เราใช้ความแม่นยำ (precision : P) การค้นคืน (recall : R) และ ตัววัด F_1 (F1-measure : F_1) ซึ่งนิยามดังนี้

$$P = \frac{\text{จำนวนตัวอย่างบวกที่ทำนายได้อย่างถูกต้อง}}{\text{จำนวนตัวอย่างที่ทำนายว่าเป็นบวก}} \quad (11)$$

$$R = \frac{\text{จำนวนตัวอย่างบวกที่ทำนายได้อย่างถูกต้อง}}{\text{จำนวนตัวอย่างทั้งหมด}} \quad (12)$$

$$F_1 = \frac{2PR}{P+R} \quad (13)$$

ผลการทดลองแสดงในตารางที่ 4 ในตารางนี้ "CoTraining (Bayes)" และ "CoTraining (Word)" เป็นผลการทดลองของตัวแยกแยะเบย์อย่างง่ายและตัวแยกแยะตัดคำของ CoTraining ตามลำดับ ส่วน "ICT (Bayes)" และ "ICT (Word)" เป็นตัวแยกแยะเบย์อย่างง่ายและตัวแยกแยะตัดคำของอัลกอริทึมการสอนไขว้แบบวนซ้ำ ตามลำดับ ส่วน U-Bayes-EM S-Bayes และ S-Word เป็นอัลกอริทึมอีเอ็ม ตัวแยกแยะเบย์อย่างง่ายแบบสอน และตัวแยกแยะตัดคำแบบสอน ตามลำดับ

ตารางที่ 5 ผลการเปรียบเทียบอัลกอริทึมสำหรับปัญหาการแยกแยะเว็บเพจภาษาไทยและไม่ใช่

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

ดังแสดงในตารางที่ 5 ICT(Word) มีประสิทธิภาพสูงสุดตามค่า F_1 ตามมาด้วย S-Bayes ตัวแยกแยะ ICT(Bayes) มีประสิทธิภาพใกล้เคียงกับ CoTraining(Bayes) ส่วน S-Word และ CoTraining(Word) มีประสิทธิภาพต่ำกว่าวิธีการอื่น ผลการทดลองแสดงให้เห็นว่า ICT ทำงานได้อย่างดีมีประสิทธิภาพทัดเทียมหรือมากกว่า S-Bayes ซึ่งเป็นอัลกอริทึมที่ใช้ตัวอย่างมีฉลากทั้งหมด และทำงานได้ดีกว่า S-Word มาก ผลการทดลองแสดงให้เห็นถึงประสิทธิภาพของ ICT ในการใช้ประโยชน์จากข้อมูลไม่มีฉลากได้อย่างดี

2.3 การจำแนกประเภทเว็บเพจออกเป็นหมวดหมู่

หัวข้อนี้กล่าวถึงการจำแนกประเภทเว็บเพจออกเป็นหมวดหมู่ โดยจะกล่าวถึงลักษณะสำคัญที่นำมาใช้ในการจำแนกเว็บเพจ ชุดข้อมูลที่นำมาทดลอง และผลการทดลองตามลำดับ

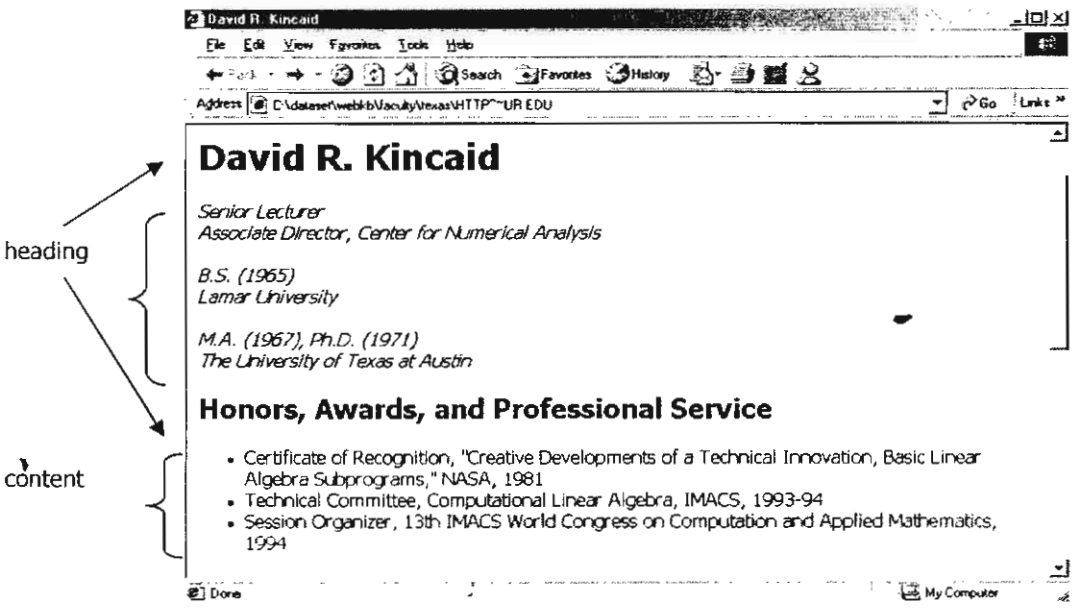
2.3.1 เซตลักษณะสำคัญ (feature set)

ในปัญหาการจำแนกเว็บเพจนั้น ประสิทธิภาพของตัวแยกแยะจะขึ้นกับเทคนิคที่ใช้ในการเรียนรู้และเซตของลักษณะสำคัญที่ใช้แทนตัวอย่างสอน เซตลักษณะสำคัญที่เหมาะสมจะช่วยให้ความถูกต้องของตัวแยกแยะสูง เซตลักษณะสำคัญที่เราเลือกใช้ในงานวิจัยนี้คือ เซตของคำที่ปรากฏในหัวข้อ (heading) ของเว็บเพจและเซตของคำที่ปรากฏในเนื้อหา (content) ของเว็บเพจ

- **เนื้อหา:** เนื้อหาของเว็บเพจให้ข้อมูลและรายละเอียดของเว็บเพจกับผู้ใช้เว็บเพจ ตัวอย่างเช่นเว็บเพจในรูปที่ 5 มีหัวข้อหนึ่งเป็น "Honors, Awards, and Professional Service" และมีเนื้อหาเป็นรายการของกิจกรรมอย่างละเอียด เราดึงเอาคำที่ปรากฏในเนื้อหาเพื่อใช้เป็นเซตของลักษณะสำคัญ ซึ่งเซตของคำในเนื้อหานี้จะใช้สำหรับสอน *Classifier2* ของอัลกอริทึมการสอนไขว้แบบวนซ้ำ
- **หัวข้อ:** สมมติฐานของเราคือคำที่อยู่ในหัวข้อจะแสดงไอเดียหลักของเนื้อหาที่ตามมา เราจึงดึงเอาคำที่ปรากฏในหัวข้อเพื่อใช้เป็นเซตของลักษณะสำคัญอีกเซตหนึ่ง และเซตของคำในหัวข้อนี้จะใช้สำหรับสอน *Classifier1* ของอัลกอริทึมการสอนไขว้แบบวนซ้ำ
- **หัวข้อและเนื้อหา:** หลังจากที่อัลกอริทึมการสอนไขว้แบบวนซ้ำเรียนรู้จบแล้ว ตัวแยกแยะผสม (combined classifier) จะทำนายฉลากหรือกลุ่ม (class) ของตัวอย่างโดยรวมเอาท์พุตจาก *Classifer1* ที่เรียนรู้จากหัวข้อ และ *Classifier2* ที่เรียนรู้จากเนื้อหา สมมติให้ $L = \{l_1, l_2, \ldots, l_m\}$ เป็นเซตของฉลาก ตัวแยกแยะผสมทำนายตัวอย่างหนึ่งๆ ว่ามีฉลากเป็น l_j ตามค่าความน่าจะเป็น $Pr(l_j | d_i)$ ในสมการด้านล่างนี้

$$Pr(l_j | d_i) = Pr(l_j | x_1)\, Pr(l_j | x_2) \qquad (14)$$

โดยที่ x_1, x_2 เป็นเซตลักษณะสำคัญจากคำในหัวข้อและคำในเนื้อหาของเอกสาร d_i ส่วน $Pr(l_j | x_1)$ และ $Pr(l_j | x_2)$ เป็นค่าความน่าจะเป็นที่เอกสารนั้นจะเป็นฉลาก l_j ที่ทำนายโดย *Classifier1* และ *Classifier2* ตามลำดับ



รูปที่ 5 หัวข้อและเนื้อหาของเว็บเพจ

2.3.2 ตัวแยกแยะย่อยของอัลกอริทึมการสอนไขว้แบบวนซ้ำและอัลกอริทึมที่นำมาเปรียบเทียบ

<u>ตัวแยกแยะย่อยของอัลกอริทึมการสอนไขว้แบบวนซ้ำ</u>

ตัวแยกแยะย่อยของอัลกอริทึมการสอนไขว้แบบวนซ้ำในกรณีนี้เหมือนกับตัวแยกแยะเบย์อย่างง่ายที่ได้อธิบายไว้ในหัวข้อที่ 2.2.1 เกือบทั้งหมด ต่างกันตรงที่ในกรณีนี้เราใช้ "*ถุงคำ (bag-of-words)*" เพื่อแทนข้อความในการทดลองในหัวข้อ 2.3.5 เราใช้ตัวแยกแยะเบย์อย่างง่ายเป็น *Classifier1* และ *Classifier2* ในตารางที่ 2

ซึ่งจะเรียนรู้จากเซตลักษณะสำคัญที่ต่างกัน โดย *Classifier1* จะเรียนจากเซตของคำในหัวข้อ ส่วน *Classifier2* จะเรียนจากเซตของคำในเนื้อหา

อัลกอริทึมที่นำมาเปรียบเทียบในการทดลอง

ในการทดลองเพื่อประเมินประสิทธิภาพของการสอนไขว้แบบวนซ้ำในปัญหาการจำแนกประเภทเวบเพจออกเป็นหมวดหมู่นั้น เราได้เปรียบเทียบกับวิธีการต่อไปนี้

(1) ตัวแยกแยะเบย์อย่างง่ายแบบสอน (Supervised Naïve Bayes Classifier – S-Bayes)

(2) อัลกอริทึมโคเทรนนิง (CoTraining Algorithm) และ

(3) อัลกอริทึมอีเอ็ม (EM Algorithm)

อัลกอริทึมโคเทรนนิงและอัลกอริทึมอีเอ็มได้กล่าวแล้วในหัวข้อที่ 2.2.2 แต่ในกรณีนี้เราใช้ *"ถุงคำ (bag-of-words)"* เพื่อแทนข้อความ

2.3.3 ชุดข้อมูลที่ใช้ในการทดลอง

เราได้นำการสอนไขว้แบบวนซ้ำไปใช้กับปัญหาการจำแนกเว็บเพจแบบหลายกลุ่ม (multi-class Web page categorization) โดยทำการทดลองกับชุดข้อมูลทั้งหมด 3 ชุดคือ

(1) ชุดข้อมูล WebKb

(2) ชุดข้อมูล WebClass และ

(3) ชุดข้อมูล DrugUsage

- ชุดข้อมูล WebKb ประกอบด้วยเว็บเพจที่เกี่ยวกับมหาวิทยาลัย ชุดข้อมูลนี้ได้จาก ftp จากมหาวิทยาลัย Carnegie Mellon University [WebKB] ชุดข้อมูลนี้ประกอบด้วยเว็บเพจทั้งหมด 981 เพจ รวบรวมจากเว็บไซต์ของ computer science department ของมหาวิทยาลัย 4 แห่ง: Cornell University, University of Washington, University of Wisconsin, และ University of Texas เว็บเพจทั้งหมดนี้ถูกติดฉลากโดยใช้แรงงานคน และถูกแบ่งออกเป็น 4 กลุ่มคือ course homepages, faculty homepages, project homepages และ student homepages มีจำนวนเว็บเพจในแต่ละกลุ่มเท่ากับ 220 เพจ 147 เพจ 81 เพจ และ 533 เพจตามลำดับ
- ชุดข้อมูล WebClass ได้มาจาก machine learning research group ในประเทศ Italy [WebClass] ชุดข้อมูลนี้เกี่ยวข้องกับงานอดิเรก ประกอบด้วยเว็บเพจจำนวน 192 เพจ แบ่งออกเป็น 4 กลุ่มคือ astronomy, jazz, auto และ motorcycle แต่ละกลุ่มมีเว็บเพจจำนวน 48 เพจ ข้อมูลในสองกลุ่มแรกค่อนข้างแตกต่างกัน ส่วนข้อมูลในสองกลุ่มหลังคือ auto และ motorcycle ค่อนข้างคล้ายกัน
- ชุดข้อมูล DrugUsage ได้มาจาก research group at Sirindhorn International Institute of Technology [DrugUsage] ชุดข้อมูลนี้ประกอบด้วยเว็บเพจจำนวน 353 เพจ แบ่งออกเป็น 5 กลุ่มในโดเมนของยา กลุ่มข้อมูลทั้งห้าคือ adverse, clinical pharmacology, overdose, patient information และ warning เว็บเพจในกลุ่ม adverse จะอธิบายเกี่ยวกับผลข้างเคียงของยา เว็บเพจในกลุ่ม clinical pharmacology จะอธิบายเกี่ยวกับการใช้ยา เว็บเพจในกลุ่ม overdose จะให้ข้อมูลเกี่ยวกับอาการผู้ป่วยที่ได้รับยาเกินปกติ เว็บเพจในกลุ่ม patient information จะเกี่ยวข้องกับข่าวสารข้อมูลสำหรับผู้ป่วยเกี่ยวกับการใช้ยา ส่วนเว็บเพจในกลุ่ม warning อธิบายคำเตือนของยาให้กับผู้ป่วย

2.3.4 การประมวลผลล่วงหน้า

ก่อนเริ่มใช้อัลกอริทึมทำการเรียนรู้นั้น เราจะทำการประมวลผลล่วงหน้าสำหรับเว็บเพจทุกเพจ โดยการลบ html tag, ตัดคำหยุด (stop word) และทำการหารากคำ (word stemming) ดังนี้

- ตัดคำหยุด: เราลบคำที่ไม่มีความสำคัญในการจำแนกเว็บเพจออกจากเพจ คำที่ถูกลบได้แก่ auxiliary verb, preposition, pronoun, possessive pronoun, phone number, digit sequence, date และ special character
- หารากคำ: คำหลายคำมีรากคำตัวเดียวกัน เช่น "teached", "teaching", "teach", "teaches", และ "teacher" ต่างก็มีรากคำเดียวกันคือ "teach" โดยการนำ word stemming มาใช้ คำทั้งห้าคำข้างต้นจะถูกแทนที่ด้วย "teach" ในการทดลองด้านล่างนี้เราใช้อัลกอริทึม Porter Stemming [Porter 1980] สำหรับการหารากคำ

2.3.5 ผลการทดลองเปรียบเทียบกับอัลกอริทึมอื่นในการจำแนกประเภทเว็บเพจออกเป็นหมวดหมู่

เราได้ทำการทดลองเพื่อวัดประสิทธิภาพของการสอนไขว้แบบวนซ้ำ โดยเปรียบเทียบอัลกอริทึมการสอนไขว้แบบวนซ้ำ (ICT) กับ อัลกอริทึมเบย์อย่างง่ายแบบสอน (Supervised Naïve Bayes – S-Bayes) ที่มีการใช้งานอย่างกว้างขวางในปัญหาการจำแนกข้อความ แต่อัลกอริทึมเบย์อย่างง่ายแบบสอนเป็นอัลกอริทึมที่ต้องอาศัยข้อมูลแบบมีฉลากทั้งหมดจึงจะทำงานได้ และอัลกอริทึมที่นำมาเปรียบเทียบอีก 2 ตัวคือ อัลกอริทึมโคเทรนนิง (CoTraining) และอัลกอริทึมอีเอ็ม (EM-algorithm) ซึ่งสามารถใช้ข้อมูลแบบไม่มีฉลากร่วมกับข้อมูลที่มีฉลากเริ่มต้นจำนวนน้อยๆ ได้เช่นกัน ในการวัดประสิทธิภาพของอัลกอริทึมนั้น เราใช้ความแม่นยำ (precision : P) การค้นคืน (recall : R) และ ตัววัด F_1 (F_1-measure) (เหมือนที่นิยามไว้ในหัวข้อที่ 2.2.3)

$$P = \frac{\text{จำนวนตัวอย่างบวกที่ทำนายได้อย่างถูกต้อง}}{\text{จำนวนตัวอย่างที่ทำนายว่าเป็นบวก}}$$

$$R = \frac{\text{จำนวนตัวอย่างบวกที่ทำนายได้อย่างถูกต้อง}}{\text{จำนวนตัวอย่างทั้งหมด}}$$

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

<u>ผลการทดลองสำหรับชุดข้อมูล WebKb</u>

การตั้งค่าต่างๆ ของการทดลองเป็นดังต่อไปนี้

- ในกรณีของอัลกอริทึมการสอนไขว้แบบวนซ้ำ เราเลือกตัวอย่างจำนวน 30% ของตัวอย่างทั้งหมดแบบสุ่มสำหรับแต่ละกลุ่ม เพื่อใช้เป็นชุดตัวอย่างสอนมีฉลากเริ่มต้น และใช้ 30% ของตัวอย่างทั้งหมดเป็นชุดตัวอย่างสอนไม่มีฉลาก ที่เหลือ 40% เป็นตัวอย่างทดสอบ
- ในกรณีของอัลกอริทึมโคเทรนนิง เราใช้ตัวอย่างสอนมีฉลากเริ่มต้นชุดเดียวกับของอัลกอริทึมการสอนไขว้แบบวนซ้ำ นอกจากนั้นชุดตัวอย่างสอนมีฉลากและชุดทดสอบก็เหมือนกับของอัลกอริทึมการสอนไขว้แบบวนซ้ำ พารามิเตอร์ p และ n ในตารางที่ 3 ถูกตั้งค่าให้มีค่าเท่ากับ 1 และ 3 ตามลำดับ
- ในกรณีของอัลกอริทึมเบย์อย่างง่ายแบบสอน เราใช้ตัวอย่างจำนวน 60% ของตัวอย่างทั้งหมดเป็นชุดตัวอย่างสอนมีฉลาก ส่วนที่เหลือ 40% เป็นตัวอย่างทดสอบ
- ในกรณีของอัลกอริทึมอีเอ็ม ชุดตัวอย่างสอนมีฉลากเริ่มต้น ชุดตัวอย่างสอนไม่มีฉลากและชุดตัวอย่างทดสอบเหมือนกันกับของอัลกอริทึมการสอนไขว้แบบวนซ้ำ

ผลการทดลองเปรียบเทียบอัลกอริทึมการสอนไขว้แบบวนซ้ำกับอัลกอริทึมอื่นๆ สำหรับชุดข้อมูล WebKb แสดงในตารางที่ 6 – ตารางที่ 9 ในการทดลอง เราใช้คำที่อยู่ในหัวข้อ (heading) ในเว็บเพจสำหรับสอน *Classifier1* และใช้คำที่อยู่ในเนื้อหา (content) ในเว็บเพจสำหรับสอน *Classifier2* ของอัลกอริทึมการสอนไขว้แบบวนซ้ำ ผลที่ได้สำหรับ *Classifier1* และ *Classifier2* แสดงด้วย Heading-based Classifier และ Content-based Classifier ตามลำดับ ส่วน Heading+content-based Classifer เป็นตัวแยกแยะที่รวมผลการแยกแยะจาก *Classifier1* และ *Classifer2* ในทำนองเดียวกันเราใช้คำที่อยู่ในหัวข้อและเนื้อหาเพื่อสอนตัวแยกแยะทั้งสองตัวของโคเทรนนิง (โคเทรนนิงใช้ตัวแยกแยะ 2 ตัวเช่นกัน) ส่วนอัลกอริทึมเบย์อย่างง่ายแบบสอนและอัลกอริทึมอีเอ็มนั้นใช้ตัวแยกแยะตัวเดียวในการเรียนรู้ ดังนั้น Heading-based Classifier และ Content-based Classifier สำหรับอัลกอริทึมทั้งสอง คือตัวแยกแยะที่ได้จากการสอนอัลกอริทึมด้วยคำในหัวข้อและคำในเนื้อหาแยกกันตามลำดับ ส่วน Heading+content-based Classifer เป็นตัวแยกแยะที่รวมผลของตัวแยกแยะเดี่ยวๆ 2 ตัวเช่นเดียวกับกรณีของการสอนไขว้แบบวนซ้ำ

ผลการทดลองในตารางที่ 6 – ตารางที่ 9 แสดงให้เห็นว่าประสิทธิภาพวัดโดยค่าเฉลี่ย F_1 ของ Heading-based Classifier ของอัลกอริทึมการสอนไขว้แบบวนซ้ำมีค่าเท่ากับ 78.25% ซึ่งสูงกว่าของอัลกอริทึมโคเทรน-นิงและอีเอ็ม ในทำนองเดียวกัน ค่าเฉลี่ย F_1 ของ Content-based Classifier ของอัลกอริทึมการสอนไขว้แบบวนซ้ำก็มีค่าสูงกว่าของอัลกอริทึมโคเทรนนิงและอีเอ็ม อย่างไรก็ดีประสิทธิภาพของการสอนไขว้แบบวนซ้ำต่ำกว่าของอัลกอริทึมเบย์อย่างง่ายแบบสอนเล็กน้อย สาเหตุที่เป็นเช่นนี้เนื่องจากอัลกอริทึมการสอนไขว้แบบวนซ้ำได้รับตัวอย่างมีฉลากเพียงครึ่งเดียวของตัวอย่างมีฉลากที่ใช้สอนอัลกอริทึมเบย์อย่างง่ายแบบสอน ส่วนตัวแยกแยะที่ใช้เซตลักษณะสำคัญทั้งสองร่วมกัน (Heading+content-based Classifier) ของอัลกอริทึมการสอนไขว้แบบวนซ้ำก็มีประสิทธิภาพสูงกว่าของอัลกอริทึมโคเทรนนิงและอีเอ็ม นอกจากนั้นเราพบว่าอัลกอริทึมการสอนไขว้แบบวนซ้ำใช้เวลาในการเรียนรู้น้อยกว่าของอัลกอริทึมโคเทรนนิงและอีเอ็ม โดยใช้เวลาประมาณ 3 นาที ส่วนอัลกอริทึมโคเทรนนิงและอีเอ็มใช้เวลามากกว่า 20 นาที ทุกอัลกอริทึมเขียนด้วยโปรแกรมภาษาจาวารันบนระบบปฏิบัติการไมโครซอฟต์วินโดวส์

ผลการทดลองสำหรับชุดข้อมูล WebClass และ DrugUsage

การตั้งค่าต่างๆ ของการทดลองสำหรับ WebClass และ DrugUsage เป็นดังต่อไปนี้

- ในกรณีของอัลกอริทึมการสอนไขว้แบบวนซ้ำ อัลกอริทึมโคเทรนนิง และอัลกอริทึมอีเอ็ม เราเลือกตัวอย่างจำนวน 33% ของตัวอย่างทั้งหมดแบบสุ่มสำหรับแต่ละกลุ่ม เพื่อใช้เป็นชุดตัวอย่างสอนมีฉลากเริ่มต้น และใช้ 33% ของตัวอย่างทั้งหมดเป็นชุดตัวอย่างสอนไม่มีฉลาก ที่เหลือ 34% เป็นตัวอย่างทดสอบ
- ในกรณีของอัลกอริทึมเบย์อย่างง่ายแบบสอน เราใช้ตัวอย่างจำนวน 66% ของตัวอย่างทั้งหมดเป็นชุดตัวอย่างสอนมีฉลาก ส่วนที่เหลือ 34% เป็นตัวอย่างทดสอบ

ผลการทดลองเปรียบเทียบอัลกอริทึมการสอนไขว้แบบวนซ้ำกับอัลกอริทึมอื่นๆ สำหรับชุดข้อมูล WebClass แสดงในตารางที่ 10 – ตารางที่ 13 ส่วนผลการทดลองสำหรับชุดข้อมูล DrugUsage แสดงในตารางที่ 14 – ตารางที่ 17 ผลการทดลองที่ได้สำหรับชุดข้อมูลทั้งสองนี้ ให้ผลในทำนองเดียวกันกับชุดข้อมูล WebKb คือ อัลกอริทึมการสอนไขว้แบบวนซ้ำมีประสิทธิภาพสูงกว่าอัลกอริทึมโคเทรนนิงและอัลกอริทึมอีเอ็ม แต่ให้ผลด้อยกว่าอัลกอริทึมเบย์อย่างง่ายแบบสอน และพบว่าประสิทธิภาพของทุกอัลกอริทึมมีค่าน้อยลงในชุดข้อมูล DrugUsage ที่เป็นเช่นนี้เนื่องจากในชุดข้อมูลนี้เว็บเพจในกลุ่มต่างๆ มีเนื้อหาที่เกี่ยวข้องกับยาทุกกลุ่ม ทำให้เนื้อหาใกล้เคียงกันกว่าชุดข้อมูลอื่นๆ ทำให้การจำแนกกลุ่มทำได้ยากขึ้น และในชุดข้อมูลนี้มีจำนวนกลุ่ม 5 กลุ่มซึ่งมากกว่าในชุดข้อมูลอื่นๆ และก็เป็นสาเหตุหนึ่งที่ทำให้การจำแนกกลุ่มยากมากขึ้นอีก

ตารางที่ 6 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมการสอนไขว้แบบวนซ้ำสำหรับชุดข้อมูล WebKb

ICT	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

ตารางที่ 7 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมเบย์อย่างง่ายแบบสอนสำหรับชุดข้อมูล WebKb

S-Bayes	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

ตารางที่ 8 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมโคเทรนนิงสำหรับชุดข้อมูล WebKb

Co Training	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

ตารางที่ 9 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมอีเอ็มสำหรับชุดข้อมูล WebKb

EM	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

ตารางที่ 10 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมการสอนไขว้แบบวนซ้ำสำหรับชุดข้อมูล WebClass

ICT	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
Motor	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

ตารางที่ 11 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมเบย์อย่างง่ายแบบสอนสำหรับชุดข้อมูล WebClass

S-Bayes	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
Motor	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

ตารางที่ 12 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมโคเทรนนิงสำหรับชุดข้อมูล WebClass

Co Training	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
Motor	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

ตารางที่ 13 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมอีเอ็มสำหรับชุดข้อมูล WebClass

EM	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
Motor	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

ตารางที่ 14 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมการสอนไขว้แบบวนซ้ำสำหรับชุดข้อมูล DrugUsage

ICT	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

ตารางที่ 15 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมเบย์อย่างง่ายแบบสอนสำหรับชุดข้อมูล DrugUsage

S-Bayes	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

ตารางที่ 16 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมโคเทรนนิงสำหรับชุดข้อมูล DrugUsage

Co Training	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

ตารางที่ 17 ประสิทธิภาพของตัวแยกแยะในอัลกอริทึมอีเอ็มสำหรับชุดข้อมูล DrugUsage

EM	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

ตารางที่ 18 – ตารางที่ 20 สรุปผลการทดลองเปรียบเทียบอัลกอริทึมการสอนไขว้แบบวนซ้ำกับอัลกอริทึมอื่นๆ สำหรับชุดข้อมูล WebKb, WebClass และ DrugUsage ตามลำดับ เราสามารถสรุปผลจากการทดลองได้ดังนี้คือ ประสิทธิภาพของอัลกอริทึมการสอนไขว้แบบวนซ้ำสูงกว่าโคเทรนนิงและอีเอ็มสำหรับทุกชุดข้อมูล ซึ่งแสดงว่าวิธีการที่นำเสนอสามารถใช้ประโยชน์จากข้อมูลที่ไม่มีฉลากได้อย่างมีประสิทธิภาพ อย่างไรก็ดี ประสิทธิภาพของการสอนไขว้แบบวนซ้ำต่ำกว่าของอัลกอริทึมเบย์อย่างง่ายแบบสอน สาเหตุที่เป็นเช่นนี้เนื่องจากตัวอย่างที่ใช้สอนอัลกอริทึมเบย์อย่างง่ายแบบสอนเป็นตัวอย่างที่ติดฉลากทั้งหมด ซึ่งต่างจากตัวอย่างที่ใช้สอนการสอนไขว้แบบวนซ้ำเป็นตัวอย่างที่ไม่มีฉลากเป็นส่วนใหญ่ ผลที่ได้โดยรวมแสดงให้เห็นว่า การสอนไขว้แบบวนซ้ำสามารถใช้งานได้อย่างดีสำหรับการจำแนกประเภทเว็บเพจ โดยรับข้อมูลที่มีฉลากเพียงเล็กน้อย สามารถใช้ประโยชน์จากตัวอย่างที่ไม่มีฉลากได้อย่างมีประสิทธิภาพ และให้ความถูกต้องสูงกว่าอัลกอริทึมที่รับข้อมูลไม่มีฉลากตัวอื่นๆ

ตารางที่ 18 ผลการทดลองเปรียบเทียบอัลกอริทึมต่างๆสำหรับชุดข้อมูล WebKb

	Heading-based Classifier			Content-based Classifier			Heading+content-based Classifier		
	P	R	F_1	P	R	F_1	P	R	F_1
ICT	71.85	94.39	78.25	67.23	86.73	71.76	73.39	88.27	76.22
S-Bayes	74.99	87.24	79.91	76.95	84.18	79.60	78.92	83.76	80.46
CoTraining	73.95	84.69	75.64	79.69	60.72	66.14	68.29	87.26	75.30
EM	68.62	91.64	75.98	76.78	74.28	70.70	74.87	78.50	70.08

ตารางที่ 19 ผลการทดลองเปรียบเทียบอัลกอริทึมต่างๆสำหรับชุดข้อมูล WebClass

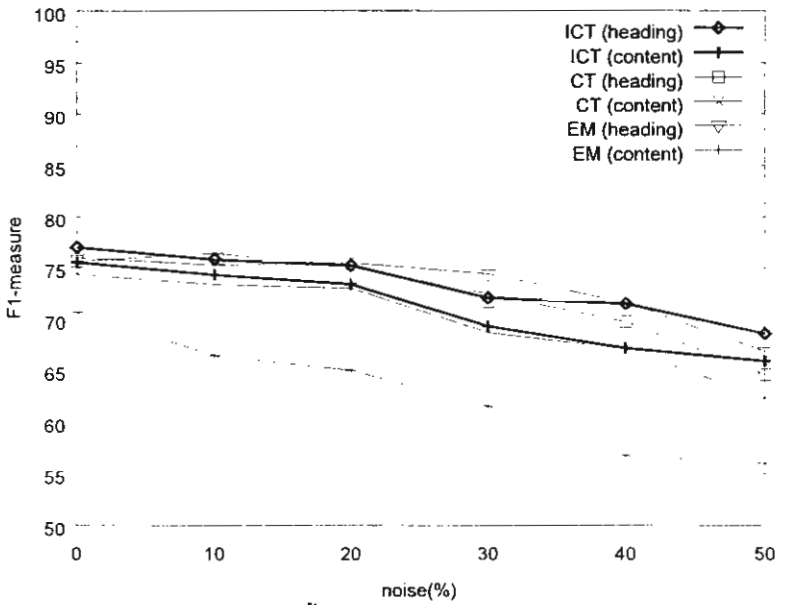
	Heading-based Classifier			Content-based Classifier			Heading+content-based Classifier		
	P	R	F_1	P	R	F_1	P	R	F_1
ICT	86.47	85.72	83.78	95.00	98.22	96.49	92.45	96.43	94.18
S-Bayes	84.36	85.12	82.17	93.14	97.62	94.99	93.97	98.21	95.81
CoTraining	74.16	92.86	80.51	83.00	98.22	89.01	87.62	97.02	91.43
EM	68.11	69.64	64.00	87.60	98.81	92.31	87.31	99.40	92.21

ตารางที่ 20 ผลการทดลองเปรียบเทียบอัลกอริทึมต่างๆสำหรับชุดข้อมูล DrugUsage

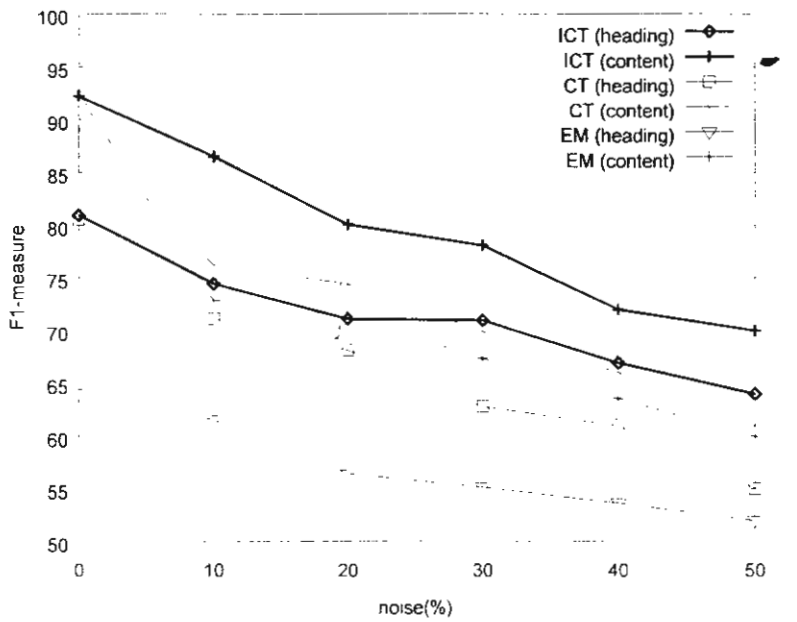
	Heading-based Classifier			Content-based Classifier			Heading+content-based Classifier		
	P	R	F_1	P	R	F_1	P	R	F_1
ICT	60.54	80.66	69.17	57.14	70.30	63.04	54.44	82.50	64.99
S-Bayes	75.74	92.67	83.35	68.81	87.12	76.89	70.57	91.63	78.47
CoTraining	55.51	62.52	58.81	50.45	77.56	61.14	52.51	79.93	61.07
EM	53.02	46.45	49.52	46.06	76.55	57.51	43.93	75.00	55.41

2.4 ผลกระทบของข้อมูลสัญญาณรบกวนในปัญหาการจำแนกเว็บเพจออกเป็นหมวดหมู่

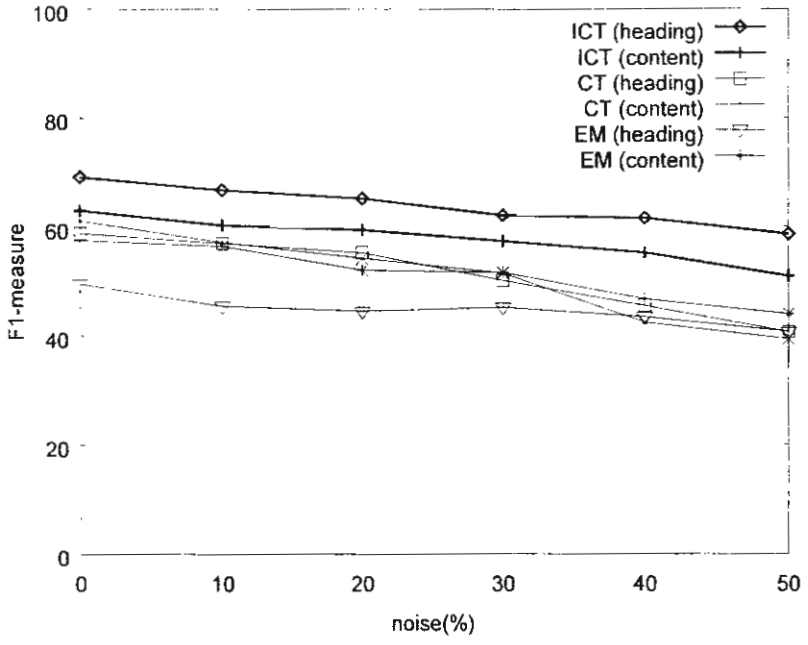
หัวข้อนี้ศึกษาผลกระทบของข้อมูลที่มีสัญญาณรบกวน (noisy data) ต่ออัลกอริทึมการสอนไขว้แบบวนซ้ำ ในการใช้งานจริง ข้อมูลสัญญาณรบกวนมักเกิดขึ้นบ่อยครั้ง อาจเนื่องมาจากการบันทึกข้อมูลที่ผิดพลาด ในกรณีของการจำแนกเว็บเพจอาจเกิดขึ้นจากความผิดพลาดของผู้สอนในการติดฉลากให้กับเว็บเพจ ดังนั้น เราจึงศึกษาว่าอัลกอริทึมจะมีประสิทธิภาพลดลงมากน้อยเพียงไรหรือมีความทนทานต่อข้อมูลสัญญาณรบกวนได้มากน้อยเท่าไร ในการทดลองด้านล่างนี้ เราเพิ่มสัญญาณรบกวนเข้าไปในข้อมูลมีฉลาก โดยการเปลี่ยนค่าของกลุ่มข้อมูลจากกลุ่มหนึ่งไปเป็นกลุ่มอื่นที่ไม่ถูกต้อง และให้สัดส่วนของข้อมูลที่มีสัญญาณรบกวนต่อข้อมูลที่มีฉลากทั้งหมดเป็น 10% ถึง 50% แล้วทำการทดลองกับชุดข้อมูลทั้งสามชุด อัลกอริทึมที่นำมาเปรียบเทียบทั้งหมดคือ อัลกอริทึมการสอนไขว้แบบวนซ้ำ อัลกอริทึมโคเทระเนิง และอัลกอริทึมอีเอ็ม ผลที่ได้แสดงในรูปที่ 6 – รูปที่ 8



รูปที่ 6 ประสิทธิภาพของอัลกอริทึมทั้งสามสำหรับข้อมูลสัญญาณรบกวนในชุดข้อมูล WebKb



รูปที่ 7 ประสิทธิภาพของอัลกอริทึมทั้งสามสำหรับข้อมูลสัญญาณรบกวนในชุดข้อมูล WebClass



รูปที่ 8 ประสิทธิภาพของอัลกอริทึมทั้งสามสำหรับข้อมูลสัญญาณรบกวนในชุดข้อมูล DrugUsage

ผลการทดลองในรูปที่ 6 – รูปที่ 8 แสดงประสิทธิภาพของอัลกอริทึมการสอนไขว้แบบวนซ้ำ (แสดงโดย ICT ในรูป) เปรียบเทียบกับอัลกอริทึมโคเทรนนิง (แสดงโดย CT ในรูป) และอัลกอริทึมอีเอ็ม (แสดงโดย EM ในรูป) ICT (heading) และ ICT (content) ในรูปแสดงผลที่ได้ของตัวแยกแยะย่อยที่ใช้เซตของคำในหัวข้อและตัวแยกแยะย่อยที่ใช้เซตของคำในเนื้อหาของอัลกอริทึมการสอนไขว้แบบวนซ้ำตามลำดับ ในทำนองเดียวกัน CT (heading) และ CT (content) เป็นตัวแยกแยะย่อยของอัลกอริทึมโคเทรนนิง ส่วน EM (heading) และ EM (content) เป็นตัวแยกแยะย่อยของอัลกอริทึมอีเอ็ม

ผลที่ได้สำหรับชุดข้อมูลทั้งสามมีลักษณะเดียวกัน คือ อัลกอริทึมการสอนไขว้แบบวนซ้ำมีความทนทานต่อข้อมูลสัญญาณรบกวนมากกว่าอัลกอริทึมอื่นๆ ซึ่งดูได้จากการลดลงของประสิทธิภาพของตัวแยกแยะในอัลกอริทึมการสอนไขว้แบบวนซ้ำที่ลดลงน้อยกว่าของอัลกอริทึมอื่นๆ ตัวอย่างเช่นในกรณีของชุดข้อมูล WebKb ในรูปที่ 6 เมื่อเราเพิ่มสัญญาณรบกวนถึง 50% ตัวแยกแยะที่ใช้คำในหัวข้อและตัวแยกแยะที่ใช้คำในเนื้อหาของอัลกอริทึมการสอนไขว้แบบวนซ้ำมีประสิทธิภาพลดลง 10.85% และ 12.70% ตามลำดับ ในขณะที่ตัวแยกแยะที่ใช้คำในหัวข้อและตัวแยกแยะที่ใช้คำในเนื้อหาของอัลกอริทึมโคเทรนนิง มีประสิทธิภาพลดลง 14.85% และ 16.17% ตามลำดับ ส่วนตัวแยกแยะที่ใช้คำในหัวข้อและตัวแยกแยะที่ใช้คำในเนื้อหาของอัลกอริทึมอีเอ็มมีประสิทธิภาพลดลง 11.99% และ 20.79% ตามลำดับ ส่วนผลสำหรับชุดข้อมูลอื่นๆ ก็ให้ผลในทำนองเดียวกัน สาเหตุหนึ่งที่อัลกอริทึมการสอนไขว้แบบวนซ้ำมีความทนทานต่อสัญญาณรบกวนมากกว่าอัลกอริทึมอื่น ก็เนื่องมาจากการที่อัลกอริทึมการสอนไขว้แบบวนซ้ำใช้การติดฉลากใหม่หมดทุกรอบการเรียนรู้จนกระทั่งการเรียนรู้ลู่เข้า ดังนั้นถ้าหากว่าในรอบหลังๆ ความถูกต้องของตัวแยกแยะมีมากขึ้น ก็จะทำให้ติดฉลากให้กับตัวอย่างไม่มีฉลากได้ถูกต้องมากขึ้นในทุกๆ รอบ แม้ว่าในรอบแรกๆ ผลของสัญญาณรบกวนอาจทำให้การติดฉลากผิดพลาด แต่ในรอบหลังๆ การติดฉลากก็สามารถทำได้ดีขึ้น ตัวอย่างเดิมที่เคยติดฉลากผิดในรอบแรกๆ ก็อาจถูกต้องได้ในรอบหลังๆ ซึ่งวิธีการนี้ต่างกับอัลกอริทึมโคเทรนนิงซึ่งใช้การติดฉลากข้อมูลไม่มีฉลากทีละน้อย และไม่มีการติดฉลากใหม่ ทำให้ผลของสัญญาณรบกวนส่งผลให้การติดฉลากผิดพลาดได้ และไม่สามารถแก้ไขได้

2.5 สรุปผลการวิจัยเทคนิคการเรียนรู้ของเครื่องและวิธีใช้ประโยชน์จากข้อมูลแบบไม่มีฉลาก

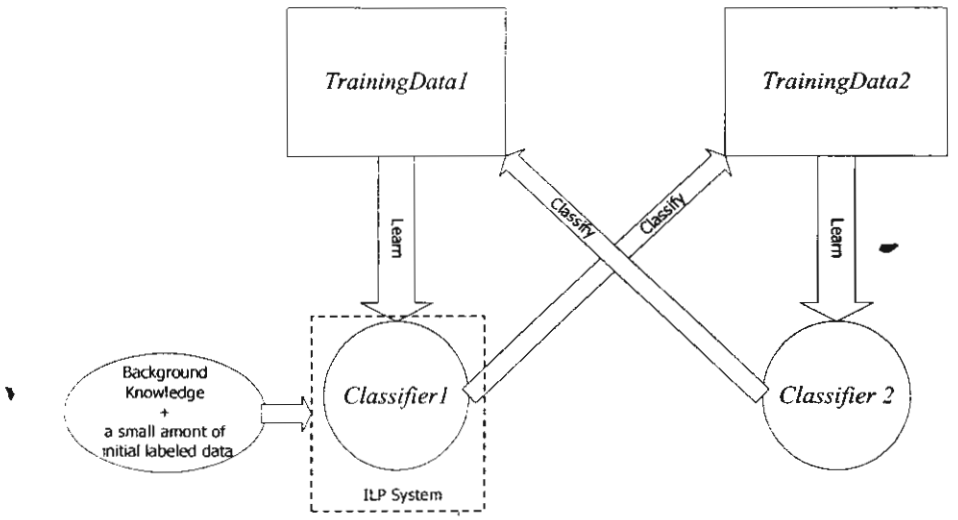
เราได้ใช้การสอนไขว้แบบวนซ้ำกับปัญหาการจำแนกประเภทเว็บเพจว่าเป็นเพจภาษาไทยหรือไม่ใช่และปัญหาการจำแนกประเภทเว็บเพจออกเป็นหมวดหมู่ ผลการวิจัยแสดงให้เห็นว่าอัลกอริทึมการสอนไขว้แบบวนซ้ำมีประสิทธิภาพใกล้เคียงกับอัลกอริทึมเบย์อย่างง่ายแบบสอน แต่มีข้อดีที่สามารถใช้ประโยชน์จากข้อมูลไม่มีฉลากได้ ทำให้จำนวนข้อมูลมีฉลากที่ต้องการน้อยลงได้ และพบว่าอัลกอริทึมการสอนไขว้แบบวนซ้ำมีประสิทธิภาพมากกว่าอัลกอริทึมอื่นๆ เช่น อัลกอริทึมโคเทรนนิง อัลกอริทึมอีเอ็ม ที่ใช้ข้อมูลไม่มีฉลากในการเรียนรู้ นอกจากนั้นจากการศึกษาถึงผลกระทบของข้อมูลสัญญาณรบกวนที่มีต่ออัลกอริทึม เราพบว่าอัลกอริทึมการสอนไขว้แบบวนซ้ำมีความทนทานต่อข้อมูลมีสัญญาณรบกวนได้ดีกว่าอัลกอริทึมอื่นๆ

3. การปรับปรุงประสิทธิภาพของการสอนไขว้แบบวนซ้ำด้วยการโปรแกรมตรรกะเชิงอุปนัย

อัลกอริทึมการสอนไขว้แบบวนซ้ำที่กล่าวในหัวข้อที่แล้วสามารถใช้จำแนกประเภทเว็บเพจและให้ผลเป็นที่น่าพอใจ อย่างไรก็ดีพบว่ายังมีข้อจำกัดในด้านประสิทธิภาพของการจำแนกประเภท ในหัวข้อนี้จะกล่าวถึงการใช้การโปรแกรมตรรกะเชิงอุปนัยมาช่วยเพิ่มประสิทธิภาพของการสอนไขว้แบบวนซ้ำ

3.1 อัลกอริทึมการสอนไขว้แบบวนซ้ำโดยใช้การโปรแกรมตรรกะเชิงอุปนัย

อัลกอริทึมการสอนไขว้แบบวนซ้ำประกอบด้วยตัวแยกแยะย่อย 2 ตัว ในหัวข้อนี้เรานำการโปรแกรมตรรกะเชิงอุปนัยมาเป็นตัวแยกแยะย่อยตัวที่หนึ่งและใช้ตัวแยกแยะเบย์อย่างง่ายเป็นตัวแยกแยะย่อยตัวที่สอง ดังแสดงในรูปที่ 9 สำหรับการโปรแกรมตรรกะเชิงอุปนัยนั้น เราได้เลือกใช้ โปรกอล (PROGOL) [Muggleton, 1995] และได้สร้างความรู้ภูมิหลังสำหรับเว็บเพจแต่ละประเภท และกำหนดเว็บเพจตัวอย่างเพื่อการเรียนรู้จำนวนหนึ่งให้กับอัลกอริทึมการสอนไขว้แบบวนซ้ำ



รูปที่ 9 อัลกอริทึมการสอนไขว้แบบวนซ้ำโดยใช้การโปรแกรมตรรกะเชิงอุปนัย

ตัวแยกแยะที่หนึ่ง (*Classifier1*) จะทำการเรียนรู้จากตัวอย่างที่มีฉลากและตัวอย่างที่ไม่มีฉลาก การทำงานจะเริ่มจากการนำความรู้ภูมิหลังและตัวอย่างที่มีฉลากมาสร้างกลุ่มของกฎ และนำกฎที่ได้มาจำแนกประเภทตัวอย่างเว็บเพจที่ไม่มีฉลาก (*TrainingData1*) หลังจากนั้นจะนำเว็บเพจที่ผ่านการจำแนกประเภทไปให้ตัวแยกแยะที่สองทำการเรียนรู้โดยใช้ตัวแยกแยะเบย์อย่างง่าย เมื่อตัวแยกแยะที่สองเรียนรู้ได้สำเร็จก็จะทำการจำแนกประเภทเว็บเพจใน *TrainingData2* ต่อมาตัวแยกแยะที่หนึ่งจะทำการเรียนรู้โดยใช้การโปรแกรมตรรกะเชิงอุปนัย และจะทำเช่นนี้ไปจนกระทั่งลู่เข้า

3.2 การสกัดลักษณะสำคัญของเว็บเพจและความรู้ภูมิหลัง

หลังจากที่ทำการตัดคำหยุดและหารากคำเรียบร้อยแล้ว เราได้ทำการสร้างเพรดิเคตเพื่อใช้เป็นลักษณะสำคัญ (feature) ให้กับระบบไอแอลพีใช้ในการเรียนรู้ เพรดิเคตที่ถูกสร้างขึ้นแบ่งเป็น 3 กลุ่มดังนี้

- เพรดิเคต *has_title(P,Word)* หมายถึง เว็บเพจ *P* มีคำ *Word* อยู่ที่ชื่อเว็บเพจ
- เพรดิเคต *has_head(P,Word)* หมายถึง เว็บเพจ *P* มีคำ *Word* อยู่ที่หัวข้อย่อยของเว็บเพจ
- เพรดิเคต *has_link(P,Word)* หมายถึง เว็บเพจ *P* มีคำ *Word* อยู่ที่ไฮเปอร์ลิงค์ของเว็บเพจ

นอกจากนี้เราได้สร้างความรู้ภูมิหลังให้แก่ประเภทของเว็บเพจโดยอยู่ในรูปของเพรดิเคต ตัวอย่างเช่น

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

เมื่อตัวแยกแยะที่หนึ่งทำการเรียนรู้เสร็จสิ้นลงจะสามารถผลิตกฎตรรกะลำดับที่หนึ่งได้ เช่น

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

ซึ่งหมายถึง เว็บเพจ *A* จะจัดอยู่ในประเภท *course homepage* เมื่อมีลิงค์ไปยังเพจ *B* และมีคำที่เกี่ยวข้องกับ *assignment* อยู่ที่ไฮเปอร์ลิงค์ซึ่งก็คือ คำว่า *project* และ *homework* เป็นต้น

3.3 การทดลอง

เราได้ดำเนินการทดลองบนข้อมูล 2 ชุด คือ WebKb และ DrugUsage โดยทำการวัดประสิทธิภาพของอัลกอริทึมโดยใช้ค่าความแม่นยำ (P) ค่าเรียกคืน (R) และ ค่า F_1 (F_1) ซึ่งนิยามไว้ในหัวข้อที่ 2.2.3 ในการทดลองด้านล่างนี้เราได้ทำการทดลองเปรียบเทียบกับอัลกอริทึมการสอนไขว้แบบวนซ้ำที่ใช้ไอแอลพี (ICT-ILP) กับอัลกอริทึมการสอนไขว้แบบวนซ้ำดั้งเดิม (ICT-NB) และยังได้ทดลองเปรียบเทียบกับอัลกอริทึมโคเทรนนิง (CoTraining) [Blum & Mitchell, 1998] อีเอ็ม (EM) [Dempster et al. 1977] และตัวแยกแยะเบย์อย่างง่ายแบบสอน (S-Bayes)

3.3.1 ผลการทดลองกับชุดข้อมูล WebKb

ในการทดลองกับชุดข้อมูล WebKb นี้ เราได้กำหนดสัดส่วนของตัวอย่างที่มีฉลากโดยสุ่มเลือกมาเป็นจำนวน 30% สำหรับเว็บเพจแต่ละประเภท และกำหนดให้ตัวอย่างในการเรียนรู้ที่ไม่มีฉลากมีจำนวน 30% ส่วนที่ใช้ในการทดสอบวัดประสิทธิภาพมีจำนวน 40% และได้ทำการทดลอง 5 ครั้งและนำผลการทดลองมาเฉลี่ย (5-fold cross validation) ผลการทดลองแสดงในตารางที่ 21

ตารางที่ 21 ประสิทธิภาพการทำงานเฉลี่ยบนชุดข้อมูล WebKb

อัลกอริทึม	ตัวแยกแยะที่ 1			ตัวแยกแยะที่ 2		
	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
S-Bayes	74.99	87.24	79.91	76.95	84.18	79.60
CoTraining	73.95	84.69	75.64	79.69	60.72	66.14
EM	68.62	91.64	75.98	76.78	74.28	70.70

ผลการทดลองพบว่าตัวแยกแยะที่หนึ่งของอัลกอริทึมการสอนไขว้แบบวนซ้ำได้ถูกพัฒนาให้มีความสามารถวัดโดยค่า F_1 เพิ่มขึ้นจากเดิม 78.25% (ICT-NB) เป็น 80.90% (ICT-ILP) และเมื่อเปรียบเทียบกับอัลกอริทึมเบย์อย่างง่ายแบบสอนซึ่งได้รับข้อมูลตัวอย่างที่มีฉลากมากกว่าอัลกอริทึมการสอนไขว้แบบวนซ้ำถึง 2 เท่า (60%) พบว่าอัลกอริทึมเบย์อย่างง่ายแบบสอนมีประสิทธิภาพด้อยกว่าอัลกอริทึมการสอนไขว้แบบวนซ้ำ (79.91%) นอกจากนี้อัลอริทึมการสอนไขว้แบบวนซ้ำชนิดการโปรแกรมตรรกะเชิงอุปนัยนี้ยังให้ประสิทธิภาพการทำงานที่สูงกว่าโคเทรนนิงและอีเอ็มอีกด้วย

เมื่อพิจารณาตัวแยกแยะที่สอง พบว่าอัลกอริทึมการสอนไขว้แบบวนซ้ำชนิดการโปรแกรมตรรกะเชิงอุปนัยสามารถเพิ่มประสิทธิภาพให้แก่ตัวแยกแยะที่สอง โดยได้ค่า F_1 เป็น 84.44% ซึ่งสูงกว่าวิธีเดิม 12.68% จะเห็นได้ว่าการโปรแกรมตรรกะเชิงอุปนัยได้มีส่วนช่วยให้ตัวแยกแยะที่สอง (ตัวแยกแยะเบย์อย่างง่าย) ทำงานได้มีประสิทธิภาพสูงขึ้น อย่างไรก็ดีอัลกอริทึมการสอนไขว้แบบวนซ้ำชนิดการโปรแกรมตรรกะเชิงอุปนัยจะใช้เวลาการเรียนรู้นานกว่าแบบเดิม

3.3.2 ผลการทดลองกับชุดข้อมูล DrugUsage

ในการทดลองกับชุดข้อมูล DrugUsage นี้ เราได้กำหนดสัดส่วนของตัวอย่างที่มีฉลากโดยสุ่มเลือกมา เป็นจำนวน 33% สำหรับเว็บเพจแต่ละประเภท และกำหนดให้ตัวอย่างในการเรียนรู้ที่ไม่มีฉลากมีจำนวน 33% ส่วนที่ใช้ในการทดสอบวัดประสิทธิภาพมีจำนวน 34% และได้ทำการทดลอง 3 ครั้งและนำผลการทดลองมาเฉลี่ย (3-fold cross validation) ผลการทดลองแสดงในตารางที่ 22

ตารางที่ 22 ประสิทธิภาพการทำงานเฉลี่ยบนชุดข้อมูล DrugUsage

อัลกอริทึม	ตัวแยกแยะที่ 1			ตัวแยกแยะที่ 2		
	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
S-Bayes	75.74	92.67	83.35	68.81	87.12	76.89
CoTraining	55.51	62.52	58.81	50.45	77.56	61.14
EM	53.02	46.45	49.52	46.06	76.55	57.51

จากผลการทดลองพบว่าอัลกอริทึมการสอนไขว้แบบวนซ้ำชนิดการโปรแกรมตรรกะเชิงอุปนัย (ICT-ILP) มีประสิทธิภาพการทำงานสูงสุดโดยเพิ่มจาก 69.17% เป็น 89.90% สำหรับตัวแยกแยะที่สองของ ICT-ILP ก็พบว่าได้ประสิทธิภาพที่สูงเมื่อเปรียบเทียบกับอัลกอริทึมอื่นๆ

3.4 สรุปผลการใช้การโปรแกรมตรรกะเชิงอุปนัยเพื่อเพิ่มประสิทธิภาพของการสอนไขว้แบบวนซ้ำ

เราพบว่าการโปรแกรมตรรกะเชิงอุปนัยได้มีส่วนช่วยให้ประสิทธิภาพของอัลกอริทึมการสอนไขว้แบบวนซ้ำทำงานได้ดียิ่งขึ้น ทั้งนี้เป็นเพราะการนำความรู้ภูมิหลังที่เหมาะสมใส่ให้แก่ระบบเพื่อใช้ในการสร้างกฎที่สามารถใช้ในการจำแนกประเภทของข้อมูลได้อย่างถูกต้อง โดยกฎที่ได้จากการเรียนรู้สามารถใช้เป็นตัวแทนของกลุ่มข้อมูลได้เป็นอย่างดีอีกทั้งยังมีความหมายที่อ่านเข้าใจได้ง่ายอีกด้วย

4. สรุปผลการวิจัย

ในงานวิจัยนี้ เราได้นำเสนอวิธีการสำหรับการจำแนกประเภทเว็บเพจออกเป็นหมวดหมู่โดยอัตโนมัติ หัวข้อที่เน้นทำวิจัย คือ (1) การวิจัยพื้นฐานเพื่อเพิ่มประสิทธิภาพของการโปรแกรมตรรกะเชิงอุปนัย และ (2) การวิจัยเทคนิคการเรียนรู้ของเครื่องที่สามารถใช้ประโยชน์จากข้อมูลแบบไม่มีฉลาก สำหรับการเพิ่มประสิทธิภาพของการโปรแกรมตรรกะเชิงอุปนัยนั้น เราได้นำเสนอวิธีการใหม่สำหรับหากฎที่ตรงกับข้อมูลมากที่สุด ซึ่งวิธีการนี้ใช้กระบวนการตึงลักษณะสำคัญร่วมกับนิวรอลเน็ตเวิร์ก ผลการวิจัยแสดงให้เห็นว่าวิธีการที่นำเสนอมีประสิทธิภาพสูงกว่าวิธีการอื่นๆ ของการโปรแกรมตรรกะเชิงอุปนัยที่นำมาเปรียบเทียบ นอกจากนี้เรายังได้นำเสนอวิธีการเรียนรู้แบบใหม่ที่เรียกว่า การสอนไขว้แบบวนซ้ำ ซึ่งสามารถใช้ประโยชน์จากข้อมูลไม่มีฉลากได้ วิธีการนี้เหมาะสำหรับการจำแนกประเภทเว็บเพจบนอินเตอร์เน็ตที่มีข้อมูลมากมายมหาศาลที่ไม่มีฉลาก การสอนไขว้แบบวนซ้ำมีจุดเด่นที่ใช้ตัวแยกแยะย่อยสองตัว ที่สามารถโต้ตอบและสอนกันเองไปมาจนกระทั่งการเรียนรู้สำเร็จ ผลที่ได้แสดงให้เห็นว่า การสอนไขว้แบบวนซ้ำสามารถใช้ประโยชน์จากข้อมูลไม่มีฉลากได้เป็นอย่างดี และมีประสิทธิภาพใกล้เคียงกับการเรียนรู้แบบสอนที่ต้องใช้ข้อมูลมีฉลากทั้งหมด เรายังได้เพิ่มประสิทธิภาพของการสอนไขว้แบบวนซ้ำ โดยการนำการโปรแกรมตรรกะเชิงอุปนัยมาเป็นตัวแยกแยะย่อย ซึ่งเพิ่มประสิทธิภาพของการสอนไขว้แบบวนซ้ำให้สูงกว่าเดิมได้อย่างมาก และมีประสิทธิภาพสูงกว่าวิธีการอื่นๆ ทุกวิธีที่นำมาทดสอบ รวมทั้งการเรียนรู้แบบสอนอีกด้วย

เอกสารอ้างอิง

Blockeel, H. and Raedt, L. D. (1997) Experiments with top-down induction of logical decision trees. Technical Report CW247. Department of Computer Sciences, K.U.Leuven.

Blum, A. and Mitchell, T. (1998) Combining labeled and unlabeled data with co-training, In *Proceeding of the Eleventh Annual Conference on Computational Learning Theory*.

Clark, P. and Niblett, T. (1989) The CN2 induction algorithm. *Machine Learning* 3(4): 261-283.

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.

Dolsak, B. and Muggleton, S. (1992) The application of inductive logic programming to finite element mesh design. In S. Muggleton (Ed.), *Inductive Logic Programming*, pp. 453-472, Academic Press.

Dzeroski, S., Schulze-Kremer, S., Heidtke, K. R., Siems, K., and Wettschereck, D. (1996) Applying ILP to diterpene structure elucidation from 13C NMR spectra. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming*.

Flach, P. and Lachiche, N. (1999) 1BC: A first-order Bayesian classifier. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming*, pp. 92-103, Springer LNAI 1634.

Joachims, T. (1998) Text categorization with support vector machines: Learning with many relevant feature, In *Proceedings of Tenth European Conference on Machine Learning*, Springer Verlag.

Kijsirikul, B. and Sinthupinyo, S. (1999) Approximate ILP rules by backpropagation neural network: A result on Thai character recognition. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming*, pp. 162-173, Springer-Verlag.

Lavrac, N. and Dzeroski, S. (1994) *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.

Lawrence, S. and Giles, C. L. (1998) Searching the World Wide Web. *Science*, 280(5360):98–100.

Ma, Y., Liu, S.W. and Huang, T.W. (1996) WWW search engines. *High Performance Communication Network Course Report, University of California at Berkeley.*

McCallum, A., Rosenfeld, R., Mitchell, T. and Nigam, A. (1998) Improving text classification by shrinkage in a hierarchy of classes, In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 350-358, Morgan Kaufmann.

Mitchell, T. (1997) *Machine Learning,* pp.180-184, McGraw-Hill. New York.

Mladenic, D. and Grobelnik, M. (1999) Assigning keywords to documents using machine learning. In *Proceedings of the Tenth International Conference on Information and Intelligent Systems IIS-99, Varazdin, Croatia.*

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

Muggleton, S. (1995) Inverse entailment and PROGOL. *New Generation Computing,* 13, 245-286.

Muggleton, S., Bain, M., Hayes-Michie, J. and Michie, D. (1989) An experimental comparison of human and machine learning algorithms. In *Proceedings of the Sixth International Workshop on Machine Learning,* pp. 113-118, Morgan Kaufmann.

Muggleton, S. and Feng, C. (1990) Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, pp. 368-381, Ohmsha, Tokyo.

Muggleton, S. and De Raedt, L. (1994) Inductive logic programming: Theory and methods, *Journal of Logic Programming*, 19:20, 629-679.

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.

Nilsson, N. J. (1980) *Principles of artificial intelligence.* Tioga, Palo Alto, CA.

Pierre, J.M. (2000) Practical issues for automated categorization of Web sites. In *Proceedings of Conference on Sematic Web.*

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.

Quinlan, J. R. (1993) *C4.5:Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.

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

Yang, Y. and Pederson, J. (1997) Feature selection in statistical learning of text categorization, In *Proceeding of the Fourteenth International Conference on Machine Learning* (pp. 412-420).

AltaVista, http://www.altavista.com

DrugUsage. http://www.kindcu.siit.ac.th

Excite, http://www.excite.com

Google, http://www.google.com

Hotbot, http://www.hotbot.com

Infoseek, http://infoseek.go.com

WebClass. http://www.di.uniba.it/~malerba/software/webclass/webclass.html

WebKb. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-51/www/co-training/data/course-cotrain-data.tar.gz

III. ผลที่ได้จากโครงการ

งานวิจัยที่ตีพิมพ์ในวารสารวิชาการระดับนานาชาติ

1. Boonserm Kijsirikul, Sukree Sinthupinyo and Kongsak Chongkasemwongse, "Approximate Match of Rules Using Backpropagation Neural Networks", Machine Learning Journal, Vol. 44 (3), pp.273-299, September, 2001.
2. Nuanwan Soonthornphisaj, Boonserm Kijsirikul, "Iterative Cross-Training: An Algorithm for Web Page Categorization", Intelligent Data Analysis, Vol. 7(3), pp.233-253, July 2003.
3. Nuanwan Soonthornphisaj, Boonserm Kijsirikul, "Iterative Cross-Training: An Algorithm for Learning from Unlabeled Web Pages", International Journal of Intelligent Systems, 2003 (to appear).

Book Chapter

4. Nuanwan Soonthornphisaj, and Boonserm Kijsirikul, "The Effects of Different Feature Sets on Web Page Categorization Problem using Iterative Cross-Training Algorithm", In Enterprise Information Systems III, J. Filipe, B. Sharp and P. Miranda (Eds.), Kluwer, pp. 132-138, 2002.

รางวัลวิจัย

ผลงานของโครงการวิจัยนี้ได้รับรางวัลวิจัยดังต่อไปนี้

5. รางวัลผลงานวิจัยดี จากกองทุนรัชดาภิเษกสมโภช ประเภทอาจารย์และนักวิจัยประจำสถาบัน ประจำปี 2545 จากจุฬาลงกรณ์มหาวิทยาลัย
6. รางวัลบทความวิจัยยอดเยี่ยม (best paper award) จาก The Second International Conference on Intelligent Technologies (InTech-2001) บทความเรื่อง "An Evaluation of the Incremental Iterative Cross-Training Approach on Web Page Classification", November 27-29, 2001.

การนำเสนอผลงานในการประชุมวิชาการนานาชาติ

7. Sukree Sinthupinyo and Boonserm Kijsirikul "Combining Neural Networks with Inductive Logic Programming for Predicting Unseen and Noisy Data", In Proc. of International Conference on Intelligent Technologies (InTech-2000), Bangkok, Thailand, December 12-14, 2000.
8. Nuanwan Soonthornphisaj and Boonserm Kijsirikul "Iterative Cross-Training: An Algorithm for Learning from Unlabeled Web Pages", In Proc. of International Conference on Intelligent Technologies (InTech-2000), Bangkok, Thailand, December 12-14, 2000.
9. Nuanwan Soonthornphisaj and Boonserm Kijsirikul, "The effects of different feature sets on Web Page Categorization Problem using Iterative Cross-Training Algorithm", International Conference on Enterprise Information Systems (ICEIS-2001), Portugal, 7-10 July, 2001.

การนำเสนอผลงานในการประชุมวิชาการนานาชาติ (ต่อ)

10. Nuanwan Soonthornphisaj and Boonserm Kijsirikul, "An Evaluation of the Incremental Iterative Cross-Training Approach on Web Page Classification", In Proc. of the Second International Conference on Intelligent Technologies (InTech-2001), Bangkok, 27-29, November 2001. (Best Paper Award).
11. Nuanwan Soonthornphisaj, Pisit Chartbanchachai, Thanapol Pratheeptham, and Boonserm Kijsirikul, "Web Page Categorization Using Hierarchical Headings Structure", International Conference on Information Technology Interface (ITI2002), Croatia, June 24-27, 2002.

การผลิตบัณฑิต

โครงการนี้ได้ผลิตบัณฑิตในระดับดุษฎีบัณฑิตและมหาบัณฑิตที่ทำวิทยานิพนธ์เกี่ยวข้องกับโครงการดังต่อไปนี้

1. นายสุกรี สินธุภิญโญ	ระดับดุษฎีบัณฑิต
2. นางนวลวรรณ สุนทรภิษัช	ระดับดุษฎีบัณฑิต
3. นายก้องศักดิ์ จงเกษมวงศ์	ระดับมหาบัณฑิต
4. นายอดุลย์ ตันธุวนิตย์	ระดับมหาบัณฑิต

ภาคผนวก

Machine Learning, 44, 273–299, 2001

Approximate Match of Rules Using Backpropagation Neural Networks

BOONSERM KIJSIRIKUL boonserm@mind.cp.eng.chula.ac.th
SUKREE SINTHUPINYO g40ssp@mind.cp.eng.chula.ac.th
KONGSAK CHONGKASEMWONGSE g41kck@mind.cp.eng.chula.ac.th
Department of Computer Engineering, Chulalongkorn University, Phayathai Rd., Phathumwan, Bangkok, 10330, Thailand

Editors: Peter Flach and Sašo Džeroski

Abstract. This paper presents a method for approximate match of first-order rules with unseen data. The method is useful especially in case of a multi-class problem or a noisy domain where unseen data are often not covered by the rules. Our method employs the Backpropagation Neural Network for the approximation. To build the network, we propose a technique for generating features from the rules to be used as inputs to the network. Our method has been evaluated on four domains of first-order learning problems. The experimental results show improvements of our method over the use of the original rules. We also applied our method to approximate match of propositional rules converted from an unpruned decision tree. In this case, our method can be thought of as soft-pruning of the decision tree. The results on multi-class learning domains in the UCI repository of machine learning databases show that our method performs better than standard C4.5's pruned and unpruned trees.

Keywords: approximate match, feature generation, inductive logic programming, backpropagation neural networks

1. Introduction

The advantages of *inductive logic programming (ILP)* (Quinlan, 1990; Muggleton, 1991; Muggleton & De Raedt, 1994) are the expressive power of first-order logic representation and the ability of employing background knowledge. ILP systems use background knowledge provided in form of first-order logic to generalize training examples, and produce rules that fit the training examples. However, when we apply an ILP system to a real-world domain, especially the domain where there are several classes of examples or the noisy domain, the produced rules may not cover or may not exactly match with the unseen data.

Consider for example the task of learning rules for the recognition of English uppercase characters. In this task, there are 26 classes of examples, i.e. 26 different English characters, and the real-world data usually contains noise such as noise due to the quality of the scanner. To classify English characters, we may use an ILP system to learn rules for each class. With exception of some systems which learr multi-class concepts (Baroglio & Botta, 1996; De Raedt & Laer, 1995; Martin & Vrain, 1995; Blockeel & Raedt, 1997), most ILP systems work with two classes of examples (positive and negative) and construct a set of rules for

the positive class. Any example not covered by the rules is classified as negative. If we want to employ these two-class systems to learn a multi-class concept, we could do this by first constructing a set of rules for the first class with its examples as positive and the other examples as negative, then constructing the sets of rules for the other classes by the same process. The learned rules are then used to classify future data, and the rule that covers or exactly matches the data can be selected as the output. One major problem of this method is that some test data, especially noisy data, may not be covered by any rule. Thus the method is unable to determine the correct rule. A commonly used technique for solving this problem is to assign the majority class recorded from training data to the test data that is not exactly matched against any rule (Clark & Niblett, 1989; Dzeroski et al., 1996).

In this paper, we are interested in solving this problem by finding the rule that provides the best match with the data. The main contribution of the paper is to present a method for the approximate match of ILP rules by using a Backpropagation Neural Network (BNN) in case of multi-class problems or noisy domains. The basic idea is that when there is no rule covering an example, we can make use of rules which *partially* match with (partially cover) the example. Some of the partially matching rules may capture important *features* (*properties*), and some may capture unimportant features of the examples. The best rule then should be the rule that matches many important features and does not necessarily match unimportant ones. The significance level of each feature is determined in terms of a weight that is trained by the BNN. Our method can deal with a related problem when a test data is covered by multiple rules.

To find the best matching rule, we also propose a novel technique for generating features from a first-order rule. The feature generation is based on the notions of *closed chains* and *open chains*. The chains define parts of rules (features) that are considered to be useful for checking properties of the examples. The features are then used by the BNN for the approximate match. We evaluate our method on four first-order datasets. The results show improvements of our method over the use of the original rules.

We also applied our method to approximate match of propositional rules converted from an unpruned decision tree. In this case, our method can be thought of as *soft-pruning* of a decision tree. Compared with standard C4.5's pruned trees on twenty datasets of multi-class learning domains in the UCI repository of machine learning databases (Merz et al., 1997), our method performs significantly better on five domains, worse on one domain and equally well on the rest.

The paper is organized as follows. Section 2 describes the method for generating features from rules and the structure of the neural network. Section 3 and 4 describe the results on first-order and propositional learning problems, respectively. Section 5 describes related and future work. Finally the conclusion is given in Section 6.

2. Approximate match of ILP rules using backpropagation neural networks

Several works have shown that combining neural networks with symbolic rules produced excellent performance (Towell & Shavlik, 1994; Mahoney & Mooney, 1994; Botta et al., 1997). In this paper, a multi-layer feedforward neural network is employed to select the rule that best matches with the input data. The algorithm for training the network used in our

method is the backpropagation algorithm (Rumelhart et al., 1986) that is widely applied to various classification problems.

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

2.1. *Feature generation*

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

First, consider a first-order rule of which every literal in the body of the rule has only variables occurring in the head. For example, consider the following rule:

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

Each literal checks a feature of an example. In such a rule, we will use each literal as a feature. We call this kind of feature *singleton feature*.

In a propositional rule, we can see that each feature examines a particular value of an attribute, such as the feature `outlook=sunny` in the following rule:

```
class=play ← outlook=sunny,humidity ≤ 75.
```

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

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

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

linked to existing variables. However, in some cases, some variables may not be in a closed chain.

Definition 2 (Open chain). A sequence of some literals in the body of a rule is said to be an *open chain* if there exists a new variable not occurring in the head of the rule which appears only once in a literal with variable(s) of the head or with variable(s) in one of the preceding literals.

Some examples of closed and open chains are shown below.

Example 1 (Closed chain). For the rule:

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

some closed chains are:

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

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

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

The definition of the open chain does not allow the variable that occurs only once in the chain but is not linked to other variables occurring in the head or in the preceding literals. For example, the open chains, for the rule in Example 1, do not include the sequences "q3(C)", "q1(A), q3(C)", "q5(D)", "q1(A), q5(D)", etc.

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

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

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

Table 1. The algorithm for feature generation.

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

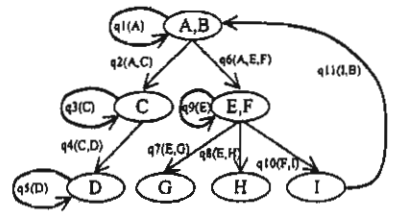


Figure 1. The dependency graph for the rule "p(A,B)←q1(A),q2(A,C),q3(C),q4(C,D),q5(D),q6(A,E,F), q7(E,G),q8(E,H),q9(E),q10(F,I),q11(I,B)."

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

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

Step 1. in Table 1

(i) q1(A)

Step 2. in Table 1

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

Step 3. in Table 1

```
 (vi) q6(A,E,F), q7(E,G)
 (vii) q6(A,E,F), q8(E,H)
(viii) q6(A,E,F), q9(E), q7(E,G)
 (ix) q6(A,E,F), q9(E), q8(E,H)
```

Step 4. in Table 1

```
(x) q6(A,E,F), q7(E,G), q8(E,H)
(xi) q6(A,E,F), q7(E,G), q8(E,H), q9(E)
```

Our method of feature generation can also be thought of as the transformation of a rule into an equivalent rule with duplicated literals that preserves the meaning of the original rule. The method reformulates the original rule into a number of parts, by using singleton, closed and open chain features, each of which is for examining a particular property of the example. For instance, using features in Example 3, the rule in Example 1 will be transformed into:

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

The obtained rule is then used to construct the structure of a neural network for enabling the approximate match, as described in the next subsection.

2.2. *Building a neural network from features*

As some ILP systems are able to specialize variables to constants and unify two variables, the head of a rule may contain constants or the same variables occurring in more than two arguments. The following rule is such an example:

```
illegal(WKf,3,WRf,WRr,BKf,WRr)←lt(WKf,3).
```

This rule contains a constant "3" at the second argument and a variable "WRr" at the fourth and sixth arguments. Before we map rules to a neural network, we first preprocess such a rule by using the unification (=) so that the head of the rule does not contain constants and the same variables. This is done by replacing constants and the same variables in the head with all different variables and adding the unifications into the body which unify the constants with the head variables and unify pairs of the head variables.

Using this preprocessing, the above rule will be converted to:

```
illegal(WKf,WKr,WRf,WRr,BKf,BKr) ← WKr=3, WRr=BKr, lt(WKf,3).
```

This preprocessing is needed because the specialization or unification can be viewed as an operator that checks for a property of the argument and thus should be considered as a feature. However, we do not replace the constant in the head if the constant defines the class. For example, the constant "1" in the rule "`mesh(A,1) ← not_important(A), not_loaded(A).`" indicates the class, and is not replaced by a variable.

Given a set of rules obtained from the preprocessing technique, we then generate the singleton, closed chain and open chain features for each rule by using the algorithm in Table 1. The features of a rule are used as input units that are linked to one hidden unit which represents the rule. Therefore, the number of hidden units in the network is the same as the number of rules. Each class is represented by one output unit of the network. In two-class problems, there are two output units, one for the positive and the other for the negative class. In multi-class problems, the number of output units is equal to the number of classes. The links from hidden units to output units are fully connected. Note that all hidden and output units include bias weights. All weights of all links and bias weights are trained by the backpropagation algorithm.

Example 4 (building a neural network from features). Consider the following rule set ($C1$, $C2$, $C3$, $C4$) in the "finite element mesh design" problem (see Section 3.2).

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

The features generated for each rule are as follows, where F_iC_j is the ith feature of the rule C_j.

```
F1C1: not_important(A)
F2C1: not_loaded(A)

F1C2: short(A)
F2C2: opposite_l(B,A)

F1C3: usual(A)
F2C3: neighbour_yz_r(A,B), cont_loaded(B)

F1C4: short(A)
F2C4: opposite_r(A,C), short(C)
F3C4: neighbour_zx_r(A,B)
```

In this example, most of the features are singleton features. F_2C_3 and F_2C_4 are closed chain features, and F_2C_2 and F_3C_4 are open chain features. A singleton feature examines a

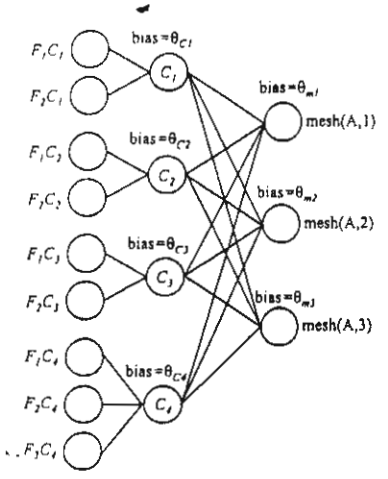


Figure 2. The structure of the neural network for the rule set $\{C_1, C_2, C_3, C_4\}$ in Example 4.

property of an example by using a single literal, such as F_2C_1: "`not_loaded(A)`" checking for the property that the edge "A" has no loading. A closed chain feature is useful for checking the property that is defined by two or more literals. F_2C_4 examines the property that an edge "C" which is opposite to "A" must be a short edge. "`Short(C)`" in F_2C_4 will be meaningless, if it is solely considered as a feature. The open chain feature F_3C_4 cannot be neglected, and it examines the property that there is some edge "B" which is a neighbour of "A".

Assume that there are only three classes, i.e. `mesh(A,1)`, `mesh(A,2)` and `mesh(A,3)`. Figure 2 shows the structure of the network for the above rules. The hidden unit C_1 representing the rule C_1 is linked to two input units each of which is a feature of C_1. Similarly, the hidden units C_2, C_3 and C_4 are linked to their features. In this example, there are three output units representing the classes.

When new variables are considered, there can be many variable bindings for a rule that make different truth values for literals containing such variables. In such a case, there can be a number of possible ways to select bindings. For example, one would use the binding that gives the minimum number of features whose truth values are true, or use bindings that assign *true* to many *important* features and few unimportant features. The use of the minimum number of *true-features* (features whose truth values are true) does not seem to be a good choice. The binding which assigns *true* to many important features and few unimportant features is a better choice. However, before the network is completely trained, it is not known in advance which features are important. Even after the network is completely trained, it is still difficult and costly to find and separate important features from unimportant ones. Therefore, in our implementation, we use a reasonable and simple method that selects the binding which gives the maximum number of features whose truth values are true. The truth value of a feature is true if the truth values of all literals of the feature are true, otherwise the truth value of the feature is false. In case there is more than one binding that gives the same value of the maximum number of true-features, we just randomly choose only one.

2.3. *Training the network*

The weights of the network are randomly initialized, and the final weights are obtained by the standard backpropagation algorithm (Rumelhart et al., 1986). In our experiment, all units in the network use the sigmoid function.

The training examples of the network are basically the same as those used to create a rule set, except for the following points.

- For multi-class problems, only examples that are covered by rules are used. As the learned rules are not representative for uncovered examples, they are not used to train the network.
- For two-class problems, all negative examples are used, and only positive examples covered by rules are used as the learned rules are representative only for the covered positive examples.[1]

Each training example is evaluated with every rule and the truth values of features are determined. We use a Prolog interpreter to evaluate the truth values of features, and thus our method works with Horn clauses including recursive rules. The features whose truth values are true are set to 1, whereas the features whose truth values are false are set to -1 for input units. The network is repetitively trained by using training examples until it converges or the number of training iterations exceeds the predefined threshold. After having been trained, the network can be used to classify unseen data. The unseen data is evaluated with features of each rule as in the training process. The truth values of features are then fed into the network, and the output with highest value will be taken as the prediction.

3. Results on first order domains

We implemented a learning system, BANNAR (Backpropagation Artificial Neural Networks for Approximating Rules). BANNAR receives a rule set, training examples and background knowledge as the inputs. The system uses the rule set to construct a feature set for building a neural network, and trains the network by using background knowledge to determine the truth values of features for each example.

In the following experiments, we selected PROGOL (Muggleton, 1995) or GOLEM (Muggleton & Feng, 1990) for learning rules. Normally we used rules produced by PROGOL as the input to BANNAR. However in our experiments on the *finite element mesh design* and the *King-Rook-King chess endgame* datasets described below, PROGOL failed to produce a rule set within a reasonable time. In those experiments, we employed GOLEM developed by the same research group of PROGOL. We then compared the results obtained by BANNAR with those of the original rule set. To show the quality of the rule set used in our method, we also included the results obtained by the other three learning systems, i.e. TILDE, 1BC and

LINUS. TILDE is a multi-class learning system that extends C4.5 to a first-order decision tree learner (Blockeel & Raedt, 1997). 1BC is a first-order probabilistic learning system using the naive Bayes algorithm (Flach & Lachiche, 1999). LINUS is a first-order learner based on the transformation approach (Lavrač & Džeroski, 1994).

In the following subsections, we briefly describe the learning systems and the datasets, and present the experimental results. Next, to understand the characteristic of BANNAR, we run additional sets of experiments and report the results in Sections 3.4, 3.5 and 3.6.

3.1. Learning systems

PROGOL

PROGOL is a state-of-the-art ILP system described in Muggleton (1995). Having positive examples, negative examples, background knowledge and *mode declarations* as inputs, PROGOL constructs a most specific rule for a random seed example. The mode declarations specify, for each argument of each predicate, the type of the argument and whether it should be a constant, a variable bound before the predicate is called, or a variable bound by the predicate. Given a most specific rule, PROGOL performs a complete search of the hypothesis space bounded below by the most specific rule, using A*-like search (Nilsson, 1980). Because of its complete search, PROGOL usually requires more time and memory than the other systems used in our experiments.

Although PROGOL is able to learn from only positive examples, the system usually works well with two-class (positive and negative) examples. Therefore, in our experiment on the *Thai character recognition* dataset (see below) which contains 77 classes of examples, a set of rules is induced for each class. Positive examples of one class are treated as negative examples of the other classes.

GOLEM

GOLEM (Muggleton & Feng, 1990) is based on the concept of relative least general generalization (rlgg) (Plotkin, 1970) with the *ij-determinate* constraint that is used to restrict the class of programs to be efficiently learned. The inputs to GOLEM are positive examples, negative examples and background knowledge. To learn a rule, GOLEM randomly selects *several* pairs of positive examples and computes the rlggs of each pair. The resulting rlgg, which is represented as a first-order rule, with the greatest coverage of positive examples is selected, and that rule is further generalized by computing the rlggs of the rule with new randomly chosen positive examples. The generalization process stops when the coverage of the best rule does not increase.

The user can specify the number of pairs to be considered for constructing rlggs (the default setting is 8). If the number of pairs is large, GOLEM will have a better chance of finding good rules. In the experiments in Section 3.3, we use the default setting. GOLEM is a two-class learning system, and needs negative examples to learn rules. In the experiment on the *finite element mesh design* dataset (see below), negative examples are generated by using the closed-world assumption.

TILDE

Unlike PROGOL and GOLEM that learn rules, TILDE learns logical decision trees (Blockeel & Readt, 1997). TILDE extends the attribute-value learner C4.5 to a first-order learning system. The learning algorithm used in TILDE is similar to C4.5. TILDE starts with the empty tree, and then generates all possible test nodes and computes the heuristic values of these test nodes. However, rather than using attribute-value tests in the nodes, TILDE employs first-order logical queries. Among the possible test nodes, it will select the one that scores best on the heuristic and place that in the current node. It will then use the partial tree to classify all examples (in the current node) to its sub-nodes. All examples passing the test will be propagated to the left, all the other ones to the right. The procedure will then recursively analyze the left and right sub-trees. When a node only contains examples of a single class (or when heuristics indicate it is uninteresting to split a node), it will be turned into a leaf. Note that TILDE is a multi-class learning system.

1BC

1BC is a first-order probabilistic learning system based on the naive Bayes algorithm. The naive Bayes algorithm works by collecting statistical information in training data, and uses this information to classify unseen data. The statistical information collected are (1) the probability of encountering each class, (2) the probability of seeing each feature given a particular class. To predict the class of an unseen data, the algorithm then finds the class that maximizes the product of the above two kinds of probabilities. See Flach and Lachiche (1999) for details. In propositional learning, a feature is a test for a specific value of an attribute, such as `attribute1=value1`. However, in first-order learning, like the problem faced in our method, 1BC has to determine what will be used as features. 1BC employs *structural predicates* and *properties* provided by users for generating *atomic features*. As the number of atomic features can be large, 1BC constrains this number by allowing the user to limit the numbers of variables and literals for each atomic feature. Atomic features containing more variables or literals than the user-specified value will not be considered. In our experiments, we asked 1BC to generate atomic features containing up to three literals and three variables. However, in the dataset of the finite element mesh design, due to memory of our computer system, we had to limit these numbers to two.

LINUS

LINUS is a first-order learning system based on the *transformation approach* (Lavrač & Džeroski, 1994). The system works by first transforming a first-order problem to an attribute-value one, and then learning with an attribute-value learner. LINUS can transform the learned description back to first-order rules. The system first transforms a first-order example with background knowledge into a set of truth-value tuples described by *features* (*literals*). Depending on background knowledge, the number of generated features may be very large. To reduce the number of generated features, Lavrač et al. (Lavrač et al., 1999) propose a technique for selecting only *relevant* features and eliminating irrelevant ones.

The attribute-value learner employed by LINUS in our experiments is C4.5 because of its availability and its multi-class learning ability. The accuracy of LINUS in our experiments is evaluated by C4.5's trees.

3.2. *Datasets*

Thai Character Recognition

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

Each example is of the form `char(A,B,C,D,E,F)`. The information contained in `A,B,C,D,E,F` are *image features*[3] extracted by a pre-processing algorithm. These image features describe various properties of a character image such as the ratio of the width and the height of the character, the structure of lines and circles that form the character, the list of zones in the images that contain junctions of lines, etc. The background knowledge contains 55 predicates. See Kijsirikul and Sinthupinyo (1999) for more details.

Finite Element Mesh Design

The dataset for the finite element mesh design (Dolsak & Muggleton, 1992) consists of 5 structures and has 13 classes (13 possible number of partitions for an edge in a structure). Each example is of the form `mesh(Edge,Number)` where `Number` indicates the number of partitions. The total number of examples is 278. The goal of finite element mesh design is to learn general rules describing how many elements should be used to model each edge of a structure. The background knowledge consists of relations describing the properties of an edge (e.g. `short`, `not_loaded`), boundary conditions (e.g. `free`), loadings (e.g. `not_loaded`), and the relations describing the structure of the object (e.g. `neighbour`). See Dolsak et al. (1994) and Dolsak and Muggleton (1992) for more details.

Mutagenesis

The dataset for the mutagenesis domain consists of 188 molecules, of which 125 are active and 63 are inactive. The goal of this problem is to predict the mutagenicity of the molecules, whether a molecule is active or inactive in terms of mutagenicity. This problem is a two-class learning problem. A molecule is described by listing its atoms `atom(AtomID,Element,Type,Charge)` and the bonds `bond(Atom1,Atom2,BondType)` between atoms. The background knowledge used in our experiment is the set $S2$ described in Srinivasan et al. (1996) that contains the definition of atom, bond, methyl groups, nitro groups, aromatic rings, hetero-aromatic rings, connected rings, ring length, and the three distinct topological ways to connect three benzene rings. See Srinivasan et al. (1996) for more details.

King-Rook-King Chess Endgame

The last dataset used in our experiment is the King-Rook-King chess endgame (KRK) dataset provided by the Machine Learning group at the University of York.[4] The goal is to learn the concept of an illegal white-to-move position with only white king, white rook and black king on the board (Muggleton et al., 1989). The number of examples in the dataset is 10,000; 3,240 representing illegal KRK endgame positions (positive), the rest representing legal endgame positions (negative). Each example is of the form `illegal(WKf,WKr,WRf,WRr,BKf,BKr)`, where `(WKf,WKr)`, `(WRf,WRr)` and `(BKf,BKr)` are the positions (file,rank) of White King, White Rook and Black King, respectively. Each of these variables takes values from 0 to 7. Background knowledge contains two relations for comparing rank and file; `adj(X,Y)` and `lt(X,Y)` indicating that rank/file `X` is adjacent to rank/file `Y` and rank/file `X` is less than rank/file `Y`, respectively.

Note that only in this dataset, for IBC we used the background relations described in Flach and Lachiche (1999), such as `board2whiteking`, `board2blackking`, `board2whiterook`, `fileeq`, `rankeq`, `pos2rank`, etc. For the other datasets described above, all background relations given to all learning systems are the same.

3.3. *Experimental results on first-order datasets*

We used three-fold cross-validation[5] and averaged the results in all experiments except for the experiment on the Thai character recognition dataset where training and test data were given. For the accuracies of PROGOL or GOLEM, we assigned the majority and the negative class to examples uncovered by the rules in the case of multi-class and two-class problems, respectively. The experimental results reporting the accuracies of all systems are shown in Table 3.[6] Table 2 reports the training times required for training BANNAR and PROGOL (GOLEM).

Table 2. The training times in seconds of BANNAR and PROGOL (GOLEM), the number of features generated by BANNAR and the number of rules generated by PROGOL (GOLEM). All systems were run on 400Mhz Pentium II. The times, the number of features and the number of rules are the average values for three-fold data, except for the TCR data set.

	BANNAR		PROGOL or GOLEM	
Data set	# Features	Time (sec.)	# Rules	Time (sec.)
TCR	467.0	39.38	77.0	1344112.92
FEM	115.0	32.98	38.0	3607.88
MUTA	18.3	12.67	6.3	6014.30
KRK	32.3	925.03	14.0	36.00

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

Data set	#Train	#Test	#Classes	BANNAR	PROGOL or GOLEM	TILDE	1BC	LINUS
TCR	1,076	2,143	77	94.40	72.00***	88.57***	77.23***	66.54***
FEM	278	3CV	13	64.45	57.80**	58.02**	46.73**	60.45*
MUTA	188	3CV	2	83.58	82.01	68.94*	77.72	74.41*
KRK	10,000	3CV	2	99.90	99.83	69.67***	87.11***	99.36***

Though our method required an additional time to train BANNAR, it was shown in Table 2 that, except for the KRK dataset, the training time of BANNAR was much less than that of PROGOL or GOLEM. PROGOL or GOLEM required quite a long time, especially on the TCR dataset. In the TCR dataset, as there are 77 classes of examples, PROGOL used a significant time to learn rules for 77 classes. The training time of BANNAR is proportional to the number of training examples and the number of rules, because it needs to match features of all rules with the examples. Therefore, in the case of the KRK dataset in which there are 10,000 examples, BANNAR used a longer time to run than in the case of the other datasets.

The results in Table 3 show that the performance of PROGOL or GOLEM was comparable to that of TILDE; PROGOL or GOLEM performed better than TILDE in two-class problems, whereas TILDE did better in multi-class problems. In the datasets tested in our experiments, 1BC did not perform well, compared to PROGOL or GOLEM. LINUS performed better than PROGOL or GOLEM only on the FEM dataset and worse on the other datasets. These results show the high accuracies of rules produced by PROGOL or GOLEM. Nevertheless, as shown in the table, BANNAR was still able to improve the accuracies of the rules, especially in the multi-class problems. Compared with the other learning systems, BANNAR performed best on all datasets. Moreover, BANNAR significantly outperformed PROGOL or GOLEM, TILDE, 1BC and LINUS on 2, 4, 3 and 4 datasets, respectively. Note that in the MUTA dataset, there are cases that multiple rules fire but there is no difficulty for BANNAR as it predicts the class which best matches the examples.

We further investigate these improvements. We want to see how well BANNAR classifies examples when they are not covered by the rules. Table 4 summarizes the results.

The results in the table show the ratio between the number of examples correctly classified and the number of examples for each portion. For example, 453/532 in the row TCR indicates that 532 examples were not covered by the rules, and 453 of them were correctly classified by BANNAR. 3/532 in the same row shows that 3 of 532 examples were correctly classified by PROGOL as we used the majority class for predicting unseen data (or the negative class for two-class problems). This means that BANNAR correctly classified 450 more examples in

Table 4. Improvements of BANNAR over the original rules, reported according to covered and uncovered examples. The columns "Covered" and "Uncovered" denote the numbers of examples covered and uncovered by the rules. Each cell denotes the number of examples correctly classified/the number of examples for that portion.

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

the case of uncovered examples. Similarly, 1570 of 1611 examples were correctly classified by BANNAR, whereas 1540 were correctly classified by PROGOL; although the number of examples covered by PROGOL was 1611, 1540 out of them were correct. A similar improvement can be seen on the FEM dataset. Slight improvements were obtained on the MUTA and the KRK datasets which are two-class problems. These results show that BANNAR significantly improved the accuracy on data which were not covered by the rules, especially in the multi-class problems.

In the next subsection, we explore if our method will help in two-class problems with the presence of noise.

3.4. *Experiments on KRK noisy datasets*

To study the effect of noise on two-class learning problems, we selected the KRK dataset. In the following experiments, three-fold cross-validation was used. The dataset was partitioned into three disjoint subsets. Each subset was used as a test set once, and the remaining subsets were used as the training set. Given training and test sets, 5%, 10% and 15% class noise was randomly added into the training set, and no noise was added into the test set. In our case, adding x% of noise means that the class value was replaced with the wrong value in x out of 100 data. For example, 5% of noise means that 5% of data were randomly selected and the class value of each data was replaced by the opposite value (from positive to negative, and vice versa). The average results of GOLEM[7] and BANNAR on noisy data are shown in Table 5. In the table, we also included the result on noise-free data for comparison.

To see how well BANNAR handles noisy data, we also ran FOSSIL on the same dataset, and reported the results together in Table 5. FOSSIL is a first-order learning algorithm that has the ability to handle noisy data by using the *correlation* heuristic to prevent learning of over-specific rules. The system has been shown to be robust against noise (Fürnkranz, 1994).

As shown in the table, when noise was added, GOLEM produced a large number of over-specific rules that were needed to cover positive training data. These rules fitted only the training data well, but they resulted in wrong classification for unseen data as shown by the decrease of the accuracy with increasing noise. The results of BANNAR show that

Table 5. The percent accuracies (Acc) of GOLEM, BANNAR and FOSSIL with 5%, 10% and 15% noise added The dataset contains 10,000 examples. The experiment was run using three-fold cross-validation. The number of rules obtained by GOLEM is the average value for three-fold data.

	GOLEM		BANNAR	FOSSIL
Noise level	#Rule	Acc	Acc.	Acc.
0%	14.00	99.83	99.90	99.02
5%	49.67	92.27	98.09	97.93
10%	134.00	87.32	98.12	96.08
15%	150.00	82.51	94.33	94.79

the accuracy decreased much slower than that of GOLEM, and BANNAR significantly improved the accuracy of GOLEM when noise was added (the differences are statistically significant at the confidence levels of 99.5% for 5% or 10% noise, and 97.5% for 15% noise). These results show BANNAR's robustness against noise which is useful to increase the accuracy of rules produced by a learning system having no ability to handle noise, such as GOLEM. Comparing the results of BANNAR with FOSSIL which has ability to handle noise, we see that the accuracies of BANNAR were comparable to those of FOSSIL. The accuracies of BANNAR were higher than those of FOSSIL in the case of 0%, 5% and 10% noise, and lower in the case of 15% noise.

The mechanism of BANNAR to prevent overfitting noisy data is different from that of FOSSIL. Whereas FOSSIL uses the heuristic to prevent learning over-specific rules, BANNAR employs neural networks to give less attention to unimportant features. Even using over-specific rules to construct a neural network, BANNAR was able to produce the network which comparatively did not overfit the data as shown by the slower decrease of its accuracy. The following example shows that BANNAR gave appropriate weights to features, i.e. higher weights to important features and lower weights to unimportant ones. In our experiment, one of rules obtained when 5% noise was added is:

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

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

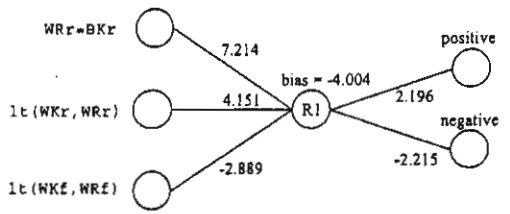


Figure 3. A portion of the network for the rule `illegal(WKf,WKr,WRf,WRr,BKf,BKr) ← WRr = BKr,lt(WKr,WRr),lt(WKf,WRf).`

-2.889 and is dominated by the sum of weights of the others which is $7.214 + 4.151 = 11.365$. Therefore, if the first two features are satisfied, the hidden unit $R1$ representing this rule will give the positive output and makes a high chance of predicting the positive class.

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

3.5. *A comparison of BANNAR's and LINUS's features*

This subsection describes experiments which compare features generated by BANNAR and those generated by LINUS. For comparison, these two sets of features were employed by the same attribute-value learners, i.e. C4.5 and a neural network. In BANNAR, as described earlier, the features were generated from rules obtained by PROGOL (GOLEM), and the truth values of the features for examples were evaluated to be used as inputs to C4.5 or a neural network. In LINUS, features were directly generated from background knowledge, and the truth values of features for examples were evaluated to be used as inputs to the same learning algorithm. In the case of C4.5, each feature generated by BANNAR or LINUS was considered as an *attribute*. In the case of a neural network, each feature was used as an input unit of the network. A neural network constructed from LINUS's features consists of three layers, i.e. input, hidden and output layers. The number of input units and the number of output units are equal to the number of features, and the number of classes, respectively. The number of hidden units was determined to be the same as the number of input units. The links from input units to hidden units, and from hidden units to output units are fully connected. The experimental results are shown in Table 6.

The results show that compared to LINUS's features, BANNAR's features gave higher accuracy, both in the case of a neural network and C4.5. This shows the effectiveness of features generated by BANNAR. Note that BANNAR generates features from rules obtained by PROGOL (GOLEM), and thus the good quality of BANNAR's features is due in part to the high accuracy of rules produced by PROGOL (GOLEM).

Table 6. The comparison of features generated by BANNAR and LINUS. The number of features or the percent accuracy (Acc.) is the average value for three-fold data, except for the TCR data set.

	# Features		Acc. of neural networks		Acc. of C4.5	
Data set	BANNAR	LINUS	BANNAR's features	LINUS's features	BANNAR's features	LINUS's features
TCR	467.0	36.0	94.40	67.10	89.22	66.54
FEM	115.0	209.0	64.45	57.97	60.46	60.45
MUTA	18.3	16.0	83.58	78.53	81.44	74.41
KRK	32.3	72.0	99.90	99.57	99.82	99.36

3.6. *Empirical study on the improved accuracy of BANNAR*

The results on previous sections show the usefulness of our method for increasing the classification accuracy of rules for unseen data, especially data in multi-class problems and noisy domains. The good performance of BANNAR is certainly due to the high accuracy of rules produced by PROGOL or GOLEM, and more improvement comes from two main elements: (1) our feature generation that enables partial match between the examples and the useful parts of rules which examine some properties of the examples, and (2) backpropagation neural networks that make use of features to enable more flexible match of rules with the examples. Though it is difficult to separately evaluate the contribution of each of these two elements to the increased accuracy of BANNAR, in this section we try to give some empirical evaluation. We designed a set of experiments that were aimed at explaining the contribution of each element to the improved accuracy of BANNAR.

The experiments were designed to evaluate that (1) how much features alone increase the accuracy, and (2) how much neural networks further increase the accuracy.

To evaluate the effect of features alone, we employ a simple technique that makes use of features to match with examples uncovered by the rules. When there are no rules which perfectly match an unseen data, we want to select the rule that provides the best match with the data. Therefore, our technique is to find the rule that contains a lot of features matching with the data. To find the best matching rule, we define $matchRatio_{r,e}$ of a rule r for an example e as follows:

$$matchRatio_{r,e} = \frac{trueFeature(r, e)}{no.\ of\ all\ features}$$

where $trueFeature(r,e)$ is the number of features of r whose truth values are true for e. The value of *matchRatio* will be high if the rule contains a lot of features matching with the example. In the case of multi-class problems, the best matching rule for the example e will be the rule that gives the highest *matchRatio*, and the corresponding class of the rule will be selected as the prediction.

However, in two-class problems, we have no rule for the negative class. Therefore, we use a simple method that will classify an example e as positive by using the following criterion.

$$\exists r \ \ matchRatio_{r,e} > \theta_r$$

where

$$\theta_r = \max_{n \in NE} matchRatio_{r,n}$$

and NE is the set of all negative training examples.

θ_r is the threshold of the rule r and is defined to be the maximum value of *matchRatio*'s among all negative training examples. The value θ_r ensures that all *negative training* examples will be classified as negative. Therefore, for an example e, if there exists $matchRatio_{r,e}$ greater than θ_r, the example will be classified as positive; otherwise it will be classified as negative.

Though it did not occur in the following experiments, in case of noisy training data, there may be some negative example that is covered by a rule; the number of true-features for that example will be equal to the number of all features. For such a rule, the maximum value of *matchRatio*'s among all negative examples will be 1. In this case, we use the weak criterion that will classify an example as positive only if *matchRatio* of that rule for the example is equal to 1 (not greater than 1).

Using this technique, in the case of multi-class problems, we evaluate *matchRatio*'s of all rules for an example to be classified, and predict as the output the class with the corresponding highest *matchRatio*. In two-class problems, we first calculate θ_r's of all rules by using the set of all negative training examples, then for an example e we evaluate $matchRatio_{r,e}$'s of all rules, and classify the example as positive if there is some $matchRatio_{r,e}$ that is greater than θ_r (or equal to 1). The results of this technique denoted as "FEATURE ALONE" are shown in Table 7. For comparison, we included the results of the original rules and the results of BANNAR in the table.

As shown in the table, the method of "FEATURE ALONE" classified more correct uncovered-examples than PROGOL or GOLEM did, especially in the multi-class problems

Table 7. The results of using features alone and those of PROGOL (GOLEM) and BANNAR on the first-order datasets, reported according to covered and uncovered examples. Each cell denotes the number of examples correctly classified/the number of examples for that portion.

		PROGOL or GOLEM (Majority or negative)		FEATURE ALONE		BANNAR	
Data set	#Test	Covered	Uncovered	Covered	Uncovered	Covered	Uncovered
TCR	2,143	1540/1611	3/532	1540/1611	222/532	1570/1611	453/532
FEM	278	146/200	14/78	146/200	24/78	145/200	34/78
MUTA	188	100/109	54/79	100/109	54/79	102/109	55/79
KRK	10,000	3350/3356	6633/6644	3350/3356	6633/6644	3352/3356	6638/6644

(TCR and FEM). For example, the number of correctly classified uncovered-examples increased from 3 to 222 and increased from 14 to 24 for the TCR and FEM datasets, respectively. This shows the effectiveness of our features for the classification of uncovered examples in multi-class problems. "FEATURE ALONE" did not help in increasing accuracy for examples covered by the rules nor for examples in two-class problems.

Comparing BANNAR to the method of "FEATURE ALONE", we can see that more uncovered-examples were correctly classified by BANNAR; the number of correctly classified uncovered-examples increased from 222 to 453 and increased from 24 to 34 for the TCR and FEM datasets, respectively. Another advantage of BANNAR over "FEATURE ALONE" was the higher accuracy for covered examples; e.g. the number of correctly classified covered-examples increased from 1540 to 1570 in the TCR dataset. All the above results show that about half of the improvement of BANNAR, which contains two main elements (features and neural networks), was due to the contribution of our features in the case of uncovered examples in multi-class problems. However, as "FEATURE ALONE" took each feature in a rule with equal significance, i.e. it simply counted the number of true-features for calculating *matchRatio*'s, it was unable to give important features higher weights than others. After neural networks were incorporated into BANNAR, more improvements were obtained as shown in the case of covered examples as well as in the case of uncovered examples.

3.7. *Summary*

In this section, we have demonstrated that BANNAR achieved good performance in classifying noisy or unseen data. The first set of experiments in Section 3.3 shows that BANNAR successfully increased the accuracy of the original rules obtained by PROGOL or GOLEM, especially on multi-class problems. After analyzing the improvements, we found that the increased accuracy was significantly obtained in the case of data uncovered by the rules. These results show that BANNAR achieves its main objective that is aimed at increasing the accuracy of rules when no rule perfectly covers the unseen data.

The higher accuracy of BANNAR compared to PROGOL and GOLEM is clearly due to the fact that noisy data or unseen data are often not covered by the rules, and the use of only the majority or negative class does not perform well. The higher accuracy of BANNAR over the other learning systems, i.e. TILDE, IBC, LINUS, is certainly due in part to the high accuracy of rules produced by PROGOL or GOLEM, and the approximate match of our method.

We next study if our method can improve the accuracy of rules in two-class problems with the presence of noise. The results in Section 3.4 confirmed the usefulness of our approximate match for increasing the accuracy of the overfitting rules. As shown by an example in figure 3, the neural network was able to give higher weights to important features and give less attention to unimportant ones. These results demonstrated the ability of neural networks which is useful for the approximate match.

The approximate match is realized by two main elements: (1) our feature generation that enables partial match between the examples and the useful parts of rules which examine some properties of the examples, and (2) backpropagation neural networks that make use

of features to enable more flexible match of rules with the examples. Though the usefulness of the approximate match was confirmed by the experiments on first-order datasets and the KRK noisy dataset, we want to know more about the contribution of each element to the increased accuracy. We then tried to give some evaluation of the contribution of these two elements by designing the additional experiments. The experimental results in Table 7 show that about half of the improvement was due to the contribution of features (the simple algorithm "FEATURE ALONE" in the table). "FEATURE ALONE" significantly increased the accuracy of the original rules in the case of uncovered examples in multi-class problems. This validates the usefulness of features constructed by our method. The disadvantage of "FEATURE ALONE" is the lack of ability to give appropriate weights to features, as it simply counts the number of true-features and all features for determining the best matching rule. Neural networks, on the other hand, have ability to give higher weights to important features, and also play an important role in the improvement.

To summarize, the good performance of BANNAR is due to two main factors: (1) the high accuracy of rules produced by PROGOL or GOLEM, and (2) the approximate match of rules with unseen or noisy data that can capture useful features and assign appropriate weights to the features by using our feature generation and backpropagation neural networks, respectively.

4. Results on propositional domains

This section presents the results of our method on propositional domains. We compare our method with C4.5 (Quinlan, 1993). For each dataset, we employ C4.5 to generate an unpruned decision tree and convert the tree to a propositional rule set. Having a rule set, we then build our BNN as described above. Our method can be thought of as *soft-pruning* of decision trees. By soft-pruning, we mean that a feature will not be completely pruned but will be given a low weight if it is not so important and a high weight if it is important.

To evaluate our soft-pruning method, we ran experiments to compare BANNAR with C4.5's unpruned and pruned trees. We also included the results of BANNAR using rules converted from C4.5's pruned trees, as we want to see how over-specific rules effect the accuracy of BANNAR. Twenty datasets of multi-class learning domains from the UCI repository (Merz et al., 1997) were used. In case of datasets where training and test sets were already provided, the results were evaluated on the given test sets. In the other datasets providing no test set, the results were averaged using 6-fold cross-validation.[8] Table 8 and Table 9 report the summary of the datasets and the classification accuracies of BANNAR and C4.5.

The results show that BANNAR was more accurate than C4.5's unpruned trees in 12 datasets, and less in 4 datasets. Compared with C4.5's pruned trees, BANNAR achieved a higher accuracy in 11 datasets and lower in 8 datasets. The results were not as good as the ones for the first-order datasets. This is because C4.5 is a multi-class learner and more importantly it can effectively deal with noisy data by pruning the overfitting trees. However, if we include confidence levels in the comparison, we can see that BANNAR performs significantly better than C4.5's unpruned and pruned trees on 6 and 5 datasets,

Table 8 The numbers of training data, test data and classes in the datasets tested in the experiments. 6CV denotes the experiment that uses six-fold cross-validation.

Data set	#Train	#Test	#Classes	Data set	#Train	#Test	#Classes
Allbp	2800	972	3	LED 17	2000	500	10
Allhyper	2800	972	5	Lymphography	148	6CV	4
Allhypo	2800	972	5	Primary-tumor	339	6CV	22
Allrep	2800	972	4	Satimage	4435	2000	6
Anneal	798	100	6	Segment	2310	6CV	7
Balance-scale	625	6CV	3	Shuttle	43500	14500	7
Glass	214	6CV	6	Soybean	307	376	19
Image	210	2100	7	Waveform	5000	6CV	3
Iris	150	6CV	3	Waveform + noise	5000	6CV	3
LED	2000	500	10	Wine	178	6CV	3

respectively. BANNAR performs significantly worse than C4.5's pruned trees only on one dataset. These results demonstrate the usefulness of our method in soft-pruning decision trees. The better results are due to more flexibility in pruning of our method: (1) converting a tree to rules before pruning will produce a non-leaf node (feature) of the tree in more than two rules, and thus this feature can be separately soft-pruned according to its participation in the rules, and (2) a feature that has some degree of significance for a rule will not be completely pruned, and will be assigned with an appropriate weight by the neural network.

Comparing the results of BANNAR with BANNAR using pruned trees, we can see that using pruned trees slightly reduced the average accuracy of BANNAR. Standard BANNAR significantly performed better than BANNAR using pruned trees on 4 datasets, and worse on 2 datasets. This shows that BANNAR performed better when it was provided with over-specific rules converted from C4.5's unpruned trees. Even when provided by rules converted from pruned trees, BANNAR still preserved its good performance.

5. Related and future work

There have been earlier learning systems which employ neural networks to improve the performance of symbolic learning. KBANN (Towell & Shavlik, 1994) and FONN (Botta et al., 1997) are the systems that use initial rules and training examples for learning neural networks. KBANN translates a propositional theory into a neural network, and uses training examples to refine the network. The results of KBANN show that its performance is better than methods which learn purely from examples. The main differences between BANNAR and KBANN are the first-order representation of the rules and the method of feature generation used in BANNAR. FONN translates first-order rules possibly including literals with new variables into a neural network. Its primary goal is to refine numerical literals of the rules. It first finds only bindings for new variables that satisfy non-numerical literals and uses these bindings for refining the numerical literals (Botta et al., 1997). The architecture

Table 9. The percent accuracies of BANNAR and C4.5 on propositional domains. The second and third columns show the accuracies of C4.5's unpruned and C4.5's pruned trees, respectively. The accuracies of BANNAR are given in the fifth column, and those of BANNAR using pruned trees are given in the fourth column. The row "Won-lost-tied (BANNAR–XXX)" shows the performance comparison between BANNAR and the corresponding systems: where XXX is C4.5 or BANNAR using pruned trees. Superscripts denote confidence levels for the difference in accuracy between BANNAR and the corresponding system, using a one-tailed paired t test: + or − is 90%, ++ or −− is 95%, +++ or −−− is 99%; no superscripts denote confidence levels that are below 90% (+ indicates higher accuracy of BANNAR, whereas − indicates lower accuracy of BANNAR). Note that the highest accuracy for each data set is shown in bold face.

Data set	C4.5 (unpruned)	C4.5 (pruned)	BANNAR + Pruned	BANNAR
Allbp	96.81	97.84	**97.94**	97.12
Allhyper	**98.87**	98.56	98.46	**98.87**
Allhypo	99.49	99.49	99.49	**99.69**
Allrep	98.77	**99.07**	**99.07**	98.97
Anneal	**97.00**	95.00	96.00	**97.00**
Balance-scale	69.76^{+++}	65.77^{+++}	$\mathbf{89.92}^{-}$	86.56
Glass	66.34	66.34	**69.11**	67.22
Image	89.43	91.00	$\mathbf{92.38}^{--}$	90.38
Iris	**95.33**	94.00	94.00	**95.33**
LED	**75.40**	75.20	74.80	74.60
LED 17	66.40	$\mathbf{75.20}^{---}$	65.00	63.80
Lymphography	73.70	78.38	**80.42**	77.72
Primary-tumor	**42.19**	41.32	33.93^{++}	38.66
Satimage	84.90^{++}	85.45	86.35	**86.80**
Segment	96.97	96.97	**97.01**	96.75
Shuttle	99.97^{+}	99.95^{++}	99.96^{++}	**99.99**
Soybean	85.64^{++}	86.70^{++}	86.44^{++}	**90.96**
Waveform	75.76^{+++}	75.82^{+++}	79.40^{++}	**80.30**
Waveform + noise	75.22^{+++}	75.30^{+++}	79.38	**79.64**
Wine	93.30	93.30	93.30	93.30
Average	84.06	84.53	85.62	85.68
Won-lost-tied (BANNAR – XXX)	12-4-4	11-8-1	10-9-1	

of FONN is based on the Factorizable Radial Basis Function network, whereas BANNAR is based on the backpropagation neural network. Another difference between BANNAR and FONN is our method of feature generation. Both KBANN and FONN as well as our system BANNAR have shown the similar results that the accuracy of rules can be increased by using neural networks.

Our method of feature generation is related to *clause templates* or *sketches* that are used as biases to define the hypothesis space in ILP systems (Bergadano & Gunetti, 1995; Tausend, 1994; Brazdil & Jorge, 1993). In MOBAL (Kietz & Wrobel, 1992), a *rule model*, which is a

kind of clause templates, defines how background predicates are used to form a clause and defines the connection of variables via the predicates. The variable connection represents how variables in a clause are linked to other variables, and is similar to our closed and open chain features.

The feature generation (selection) has been addressed in earlier ILP systems, such as LINUS and 1BC. LINUS generates features for transforming a first-order example into a propositional representation in form of truth-value tuples described by the features. Depending on background knowledge which is used to generate features, the number of features can be very large. Lavrač et al. (Lavrač et al., 1999) describe a method for selecting only relevant features in order to reduce the hypothesis space searched by LINUS. The method is based on the *discriminating power* of a feature. Feature selection in LINUS and feature generation in BANNAR are employed for different purposes. The main purpose of feature selection in LINUS is for reducing the hypothesis space employed by an attribute-value learner, whereas our aim of feature generation is to generate a sequence of literals for partial matching with unseen examples. 1BC employs *structural predicates* and *properties* provided by users for generating *atomic* features. The features are then used to train the naive Bayes classifier for learning statistical information from training examples. The difference between our feature generation and that of 1BC is that our method does not require the user to give additional information for generating the features.

Both LINUS and 1BC generate features *directly* from background knowledge. However, our method needs rules obtained from PROGOL or GOLEM to generate features. This is a limitation of our method that cannot directly generate features from background knowledge. One of our future research plans is to extend the feature generation to a more powerful technique that can directly generate features from background knowledge. Another interesting research direction is to combine a naive Bayes classifier, like one in 1BC, with our method that generates features from previously learned rules, and use these features to learn statistical information for the classifier.

Another disadvantage of our method is that the learned neural networks are very difficult to understand to the user. We plan to study a method that transfers the networks into a more understandable representation in the future.

Another direction for our future research is to investigate a more sophisticated method for evaluating the truth values of features, such as fuzzy logic. In the current work, if truth values of some literals of a feature are false, the truth value of the whole feature will be false. If we can assign a more suitable value, it may increase the classification accuracy.

There are some ILP systems that are able to learn multi-class concepts represented in first-order rules, such as RTL (Baroglio & Botta, 1996), ICL (De Raedt & Laer, 1995), MULT_ICN (Martin & Vrain, 1995). Nevertheless, it is still possible that some unseen data may not exactly match the rules produced by these systems. We believe that our method can also be applied to these systems.

6. Conclusions

We have proposed a method for approximate match of rules using neural networks. The main contribution of our work is the method of using feature generation and neural networks to increase the accuracy of first-order rules in case of multi-class problems or noisy domains where unseen data are often not covered by the rules. We have also proposed a novel technique to generate first-order features to be used for the approximate match of rules. Our method has been evaluated on four domains of first-order learning problems. The experimental results show that our method gives significant improvements over the use of the original rules. The improved results come from the combination of the ILP system and our system BANNAR. The ILP system produces rules that accurately classify the training data, and BANNAR makes the rules more flexible for partially matching with unseen or noisy data. We also applied BANNAR to approximate match of propositional rules converted from an unpruned decision tree. The results show that our method performs better than standard C4.5's pruned and unpruned trees.

Acknowledgments

We would like to thank Prabhas Chongstitvatana, Apinetr Unakul and Surapant Meknavin for comments on the paper. We are grateful to the anonymous referees for useful comments and suggestions that improved this work. This work is supported by the Thailand Research Fund. The views and conclusions contained in this paper are those of the authors and the Thailand Research Fund is not necessarily of the same opinion.

Notes

1. We also ran experiments by using *all* positive and all negative examples, and found that the results are almost the same.
2. The dataset will be made available at http://mind.cp.eng.chula.ac.th.
3. These are features describing the structure of the character images, such as the ratio of the width and the height of the character image. These features should not be confused with the feature generation described in Section 2.1.
4. http://www-users.cs.york.ac.uk/~stephen/chess.html
5. As training ILP systems requires quite a long time, we used only three folds in the cross-validation experiments.
6. The results were obtained by using the default settings of all systems.
7. In the experiments below, the number of pairs of examples to be considered for constructing rlggs is set to 50.
8. Compared to the first-order domains which require long time to train ILP systems, C4.5 ran fast in these propositional domains, and thus instead of 3-fold we used 6-fold cross-validation in the experiments.

References

Baroglio, C., & Botta, M. (1995). Multiple predicate learning with RTL. In *Proceedings of the Fourth Congress of the Italian Association for Artificial Intelligence, (AI*IA95)* (pp. 44–55). Florence, Italy.

Bergadano, F., & Gunetti, D. (1995). *Inductive logic programming: From machine learning to software engineering*. Cambridge, MA: The MIT Press.

Blockeel, H., & Raedt, L. D. (1997). Experiments with top-down induction of logical decision trees. Technical Report CW247, Department of Computer Science, K.U. Leuven.

Botta, M., Giordana, A., & Piola, R. (1997). FONN: Combining first order logic with connectionist learning. In *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 48–56). San Francisco, CA: Morgan Kaufmann.

Brazdil, P., & Jorge, A. (1993). Exploiting algorithm sketches in ILP. In *Proceedings of the Third International Workshop on Inductive Logic Programming*. Ljubljana, Slovenia: Jozef Stefan Institute.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*(4), 261–283.

Dolsak, B., Jezernik, A., & Bratko, I. (1994). A knowledge base for finite element mesh design. *Artificial Intelligence in Engineering*, *9*, 19–27.

Dolsak, B., & Muggleton, S. (1992). The application of inductive logic programming to finite element mesh design. In S. Muggleton (Ed.), *Inductive logic programming*. London: Academic Press.

Džeroski, S., Schulze-Kremer, S., Heidtke, K. R., Siems, K., & Wettschereck, D. (1996). Applying ILP to diterpene structure elucidation from 13C NMR spectra. In *Proceedings of the Sixth International Workshop on Inductive Logic Programming* (pp. 14–27). Lecture notes in artificial intelligence (Vol. 1314). Berlin: Springer-Verlag.

Flach, P., & Lachiche, N. (1999). 1BC: A first-order Bayesian classifier. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming* (pp. 92–103). Lecture notes in artificial intelligence (Vol. 1634). Berlin: Springer-Verlag.

Fürnkranz, J. (1994). Fossil: A robust relational learner. In *Proceedings of the Seventh European Conference on Machine Learning, ECML-94* (pp. 122–137). Berlin: Springer-Verlag.

Kietz, J. U., & Wrobel, S. (1992). Controlling the complexity of learning in logic through syntactic and task-oriented models. In S. Muggleton (Ed.), *Inductive logic programming*. London: Academic Press.

Kijsirikul, B., & Sinthupinyo, S. (1999). Approximate ILP rules by backpropagation neural network: A result on Thai character recognition. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming* (pp. 162–173). Berlin: Springer-Verlag.

Lavrač, N., & Džeroski, S. (1994). *Inductive logic programming: Techniques and applications*. New York: Ellis Horwood.

Lavrač, N., Gamberger, D., & Jovanoski, V. (1999). A study of relevance for learning in deductive databases. *Journal of Logic Programming*, *40*, 215–249.

Mahoney, J., & Mooney, R. (1994). Comparing methods for refining certainty-factor rule-bases. In *Proceedings of the Eleventh International Workshop on Machine Learning*. (pp. 173–180). New Brunswick, NJ.

Martin, L., & Vrain, C. (1995). MULT_ICN: An empirical multiple predicate learner. In *Proceedings of the Fifth International Workshop on Inductive Logic Programming* (pp. 129–144). Department of Computer Science, Katholieke, Universiteit Leuven.

Merz, C. J., Murphy, P. M., & Aha, D. W. (1997). UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, CA. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing*, *8*(4), 295–318.

Muggleton, S. (1995). Inverse entailment and PROGOL. *New Generation Computing*, *13*, 245–286.

Muggleton, S., Bain, M., Hayes-Michie, J., & Michie, D. (1989). An experimental comparison of human and machine learning algorithms. In *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 113–118). Los Altos, CA: Morgan Kaufmann.

Muggleton, S., & Feng, C. (1990). Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory* (pp. 368–381). Tokyo: Ohmsha.

Muggleton, S., & Raedt, L. D. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, *19*(20), 629–679.

Nilsson, N. J. (1980). *Principles of artificial intelligence*. Palo Alto, CA: Tioga.

Plotkin, G. D. (1970). A note on inductive generalization. In B. Meltzer & D. Michie (Eds.), *Machine intelligence* (Vol. 5). New York: Elsevier North-Holland.

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

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Raedt, L. D., & Laer, W. V. (1995). Inductive constraint logic. In *Proceedings of the Fifth Workshop on Algorithmic Learning Theory* (pp. 80–94). Lecture notes in artificial intelligence (Vol. 997). Berlin: Springer-Verlag.

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

Srinivasan, A., Muggleton, S., Sternberg, M. J. E., & King, R. D. (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, *85*, 277–299.

Tausend, B. (1994). Representing biases for inductive logic programming. In *Proceedings of European Conferences on Machine Learning, ECML-94* (pp. 427–430). Berlin: Springer-Verlag.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, *70*(4), 119–116.

Received May 26, 2000
Revised August 29, 2000
Accepted December 20, 2000

Intelligent Data Analysis 5 (2001) 1–21 IDA128
IOS Press

Iterative cross-training: An algorithm for web page categorization

Nuanwan Soonthornphisaj[a,*] and Boonserm Kijsirikul[b]
[a]*Machine Intelligence and Knowledge Discovery Laboratory, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand, 10330*
Tel.: +661 6592877; Fax: +662 2186955; E-mail: Nuanwan@chula.com
[b]*Home address: 100/693 Chollada 48A, Bangkruay-Sainoi Rd. Bangbuathong, Nonthaburi, Thailand, 11110*
[c]*Machine Intelligence and Knowledge Discovery Laboratory, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand, 10330*
Tel.: +661 6592877; Fax: +662 2186955; E-mail: boonserm.k@chula.ac.th

Received 24 July 2002
Revised 27 August 2002
Accepted 18 September 2002

Abstract. The goal of 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 not so easy 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 a novel approach called *Iterative Cross-Training* (*ICT*). In this paper, we applied the algorithm to Web page categorization on three data sets. The performance of ICT was evaluated and analyzed with the supervised learning algorithms, Co-Training and Expectation Maximization. We found that ICT is considered to be an effective approach for the Web page categorization task.

Keywords: Machine learning, web mining, web page categorization

1. Introduction

The availability of large, heterogeneous repositories of Web pages is increasing rapidly. There are billions of Web pages accessible on the Internet with 1.5 million pages being added daily [9]. 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 a Yahoo, [1] Google[2] etc., organize their Web resources in category-specific style. These Web sites currently employ human experts to categorize the documents. However, the number of Web pages nowadays is exponentially increasing.

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

It is difficult to keep updating and maintaining the index of billions of Web pages. To improve category specific search, we need a well-trained classifier with a high ability to recognize Web pages of a specified category.

Many traditional machine learning techniques have been investigated and applied to the Web page categorization problem. The algorithm called bootstrapping was investigated in the domain of text learning by Jones et al. [4]. This algorithm requires knowledge about the classes of documents, which is provided in the form of a few keywords per class and a class hierarchy. The algorithm proceeds by using the keywords to generate preliminary labels for some documents by term matching. Then these labels, the hierarchy and all of unlabeled documents become the input to a bootstrapping algorithm. The bootstrapping algorithm combines two techniques, which are the hierarchy shrinkage and Expectation-Maximization (EM) with unlabeled data. The algorithm was tested with the topic identification of computer science research papers, and the experimental result was that the algorithm obtained 66% correctness.

The Co-Training algorithm was first introduced in [1]. The concept of the algorithm is based on the boosting technique. This means that, the algorithm learns from a small amount of initial labeled data, and then incrementally classifies unlabeled data into categories. The basic assumption of Co-Training is that the instance distribution is compatible with the target function. It requires that for most examples, the target functions over each feature set predict the same label. For example, in the Web page domain, the class of the instance should be identifiable using either the hyperlink text or the page text alone. The second assumption is that the features in the first feature 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 results show that the Co-Training algorithm achieves quite good performance in a page-based classifier and a hyperlink-based classifier.

Another interesting technique is Support Vector Machines (SVMs), which was applied to Web page categorization on the WebKb data set. Joachims studied the effects of SVMs using different kernel functions [3]. He used one kernel to represent a document base on words appearing in the content, and the other kernel to represent the words appearing in the hyperlinks. The results led to the conclusion that the combination of two kernels gave good performance on text categorization problem.

The Expectation-Maximization (EM) is another boosting style algorithm, which was first introduced by Dempster et al. [2]. It is an iterative algorithm for maximum likelihood estimation in problems with incomplete data. Given a model of data 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: the E-step and M-step. The E-step calculates probabilistically weighted class labels for every document using the classifier. The M-step estimates new classifier parameters using all documents. Nigam et al.'s work [8] combined EM with a naive Bayes classifier to solve the text classification problem. The algorithm has been shown to be able to significantly increase text classification accuracy when given limited amounts of labeled data and large amounts of unlabeled data.

This paper proposes a new learning algorithm called Iterative Cross-Training (ICT), which consists of two classifiers that co-operate each other during the learning process. The algorithm requires only a small amount of initial labeled data when no domain knowledge is given. If domain knowledge is supplied to the algorithm, ICT does not require any initial labeled data. Experimental results show that its performance is comparable to Co-Training and EM, and is more robust to noise data. Moreover, ICT consumes less computational time compared to Co-Training and EM.

The remainder of the paper is organized as follows. Section 2 describes the ICT algorithm in detail. Section 3 presents other learning algorithms in detail. Section 4 presents experimental results and Section 5 is the conclusion.

2. Iterative cross-training algorithm

The motivation of the 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 can simply employ *bag-of-characters* representation (see Section 2.1.2). Therefore hypotheses, working within the first representation language, promise to be more efficient, but have more computational cost than those working with the second one. On the other hand, those working within the second representation language are likely to be less accurate, but consume less computational time (because they need no domain knowledge). Therefore we propose in this paper a novel approach that combines these two representations.

This is the idea of ICT, which actually learns simultaneously from two sets of hypotheses, one in each language. This can be formalized as follows:

Given:

- a set of labeled and unlabeled examples LE and 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.

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 1 shows 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.

When an active classifier labels data, it can ask for the confirmation from the other classifier to make a decision about which class the example should be. If both classifiers agree with the same classification

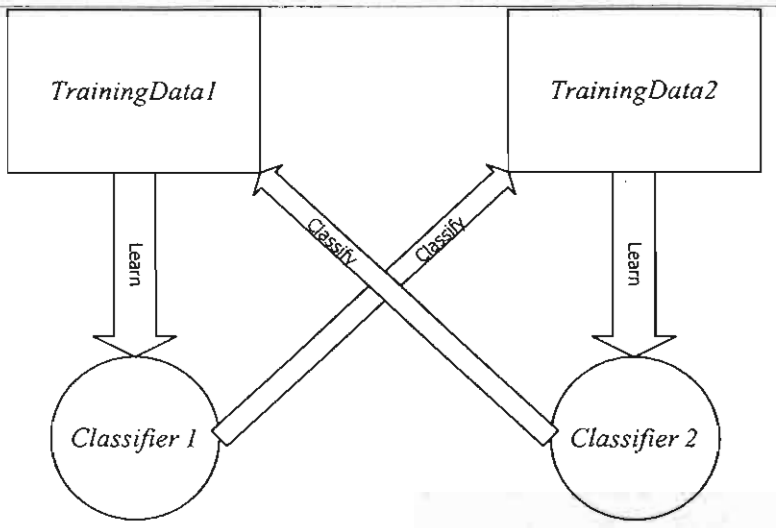


Fig. 1. Iterative Cross-Training algorithm.

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. The purpose of the consistency checking is to produce more reliable labeled data, but the checking will slow down the learning process.

In order to see the empirical behaviour of ICT, we conduct two types of experiments as follows.

(1) A Web page classification problem when domain knowledge is available (Thai/non-Thai Web page classification).
(2) Web page classification problems when no domain knowledge is given (Web page categorization problems on WebKb and WebClass data sets).

2.1. *Sub-classifiers in ICT for the classification of thai/non-thai web pages*

In the problem of classification of Thai/Non-Thai Web pages, our goal is to classify Web pages into Thai and non-Thai pages. This problem is of our interest because we want to build a Web robot that efficiently crawls the Web and retrieves only Thai pages for building a Thai search engine. In this problem, the first sub-classifier, *Classifier1*, is given some knowledge about the domain in the form of a dictionary and uses the dictionary to help in determining whether a page is written in Thai or not. The algorithm used by *Classifier1* is the word segmentation algorithm that will be described below. The second sub-classifier, *Classifier2*, is given no knowledge and uses the naive Bayes classifier. The *Classifier1* is the word segmentation classifier, which has a very high potential in identifying the Thai language. Unfortunately, it consumes much more computational time than a simple naive Bayes classifier. Therefore, we employed ICT in order to let the stronger classifier boost the performance of the weaker classifier. Then the boosted classifier can be applied to solve the problem, instead.

2.1.1. *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

Fig. 2. All possible word segmentations of a Thai sentence.

is in that language. We cannot expect that all words in the page appear in the 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 [6].

Note that a string of Thai characters can usually be segmented in many possible ways as shown in Fig. 2 because a word may be a substring of a longer word, and without a word delimiter it is difficult to find which segmentation is correct. Below we describe the method for word segmentation. Given a Thai dictionary and a document d of n characters $(c_1, c_2, \ldots, c_n)$, the word segmentation classifier generates all possible segmentations and finds the best segmentation $(w_1, w_2, \ldots, w_n)$ that minimizes the cost function in Eq. (1)

$$\arg \min_{w_i,\cdots,w_n} \sum_{i=1}^{m} \text{cost}(w_i) \tag{1}$$

where

$$\text{cost} = \begin{cases} \eta_1 \text{ if } w_i \text{ is a word in the dictionary} \\ \eta_2 \text{ if } w_i \text{ is a word in the dictionary} \end{cases}$$

In the following experiments, η_1 and η_2 are set to 1 and 2, respectively. As generating all possible segmentations and calculating their costs is very expensive, we employ dynamic programming technique to implement this calculation. Note that any sequence of characters, $c_i, \ldots, c_j$, found in the dictionary must be considered as a word, and must not be grouped with nearby characters to form a long unknown string.

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

$$WordRatio = \frac{\text{the number of characters in all words}}{\text{the number of all characters in the document}} \tag{2}$$

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 (θ_1).

Having only the threshold of *WordRatio* (θ_1) as the parameter, we can find θ_{10} that produces more true positive and true negative examples for *TrainningData2*. As described above, most Thai pages should have a high value of *WordRatio*, whereas non-Thai pages should have a low value. If the numbers of

Thai and non-Thai pages in *TrainningData2* are the same, it is easy to see that any value of θ_{10} will give more correctly classified pages than incorrectly ones (except for $\theta_{10} = 0.0$ or $\theta_{10} = 1.0$, 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 θ_{10}, (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.

A new θ_1 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 $SP \geqslant LN$, the new θ_1 is estimated as:

$$\theta_1 = \frac{SP + LN}{2} \tag{3}$$

Now, consider the case of $SP < LN$. Let $V_1 = SP$, V_n=LN, and $V_2, \ldots, V_{n-1}$ be the values between V_1 and V_n ($V_1 \leqslant V_2 \leqslant V_{n-1} \leqslant V_n$). The new θ_1 is estimated as:

$$\theta_1 = \frac{V_{i^\bullet} + V_{i^\bullet+1}}{2} \tag{4}$$

$$V_{i^\bullet} = arg \min_{V_i}(\text{no. of } V_j + \text{no. of } V_k) \tag{5}$$

Where V_k is a value of a labeled positive example, V_j is a value of a labeled negative example, and $V_1 \leqslant V_k \leqslant V_i$, $V_{i+1} \leqslant V_j \leqslant V_n$. If SP is greater than LN, θ_1 will completely separate the labeled positive from negative examples. Otherwise, θ_1 will give the minimum errors of misclassified examples.

2.1.2. *Naive bayes classifier* (*Classifier2*)

For text classification, naive Bayes is among the most commonly used and the most effective methods [7]. 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 $Pr(l_j \mid c_1, \ldots, c_n)$:

$$l^* = arg \max_{l_j} Pr(l_j \mid c_1, \cdots, c_n) \tag{6}$$

$$= arg \max_{l_j} \frac{Pr(c_1, \cdots, c_n \mid l_j) Pr(l_j)}{Pr(c_1, \cdots, c_n)} \tag{7}$$

$$= arg \max_{l_j} Pr(c_1, \cdots, c_n \mid l_j) Pr(l_j)$$

$$l^* = arg \max_{l_j} Pr(l_j) \prod_{i=1}^{n} Pr(c_i \mid l_j, c_1, \cdots, c_{i-1})$$

$$= arg \max_{l_j} Pr(l_j) \prod_{i=1}^{n} Pr(c_i \mid l_j)$$

In this case, L is the set of positive and negative class labels. The term $Pr(c_1,\cdots,c_n)$ in Eq. (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 $d = (c_1,c_2,\cdots,c_n)$, calculating the term $Pr(c_1,c_2,\cdots,c_n \mid l_j)$ 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 [7]. 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 [5,7,14]. Using the above assumptions, Eq. 7 can be simplified to Eq. 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 $Pr(c_i \mid l_j)$ are used as the parameter set 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 $Pr(c_i \mid l_j)$, of seeing character c_i given class label l_j, is estimated by the following equation:

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

where $N(c_i,l_j)$ is the number of times character c_i appears in the training 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 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.

2.2. *Sub-classifiers in ICT for the web page categorization task*

For the problem of Web page categorization, the algorithm employs bag-of-words representation. In this problem, each Web page contains two sets of features as follows:

(1) words appearing in the content of the page (this feature will be used by the content-based classifier),
(2) words appearing in the headings in that page (this feature will be used by the heading-based classifier).

Therefore, each page can be viewed in two different ways, i.e. , content-based features and heading-based features. With these two feature sets, we construct two naive Bayes classifiers; the first one (*Classifier1* in Table 1) learns its model from heading-based features and the second one (*Classifier2*) learns from content-based features. Both classifiers use the naive Bayes algorithm, which is the same algorithm described in the previous subsection except that in this case we employ *bag-of-words* representation.

3. Classifiers used in comparison

In order to evaluate the effectiveness of ICT, we compared our algorithm with Expectation-Maximization, Co-Training and supervised learning algorithms. This section provides some background information of these algorithms.

Table 1
The training algorithm of iterative cross-training

Given:
two sets $TrainingData1$ and *TrainningData2* of unlabeled training examples.
Initialize the parameter set of *Classifier1* to θ_{10}
$\theta_1 \rightarrow \theta_{10}$
Initialize the parameter set of *Classifier2* to θ_{20}
$\theta_2 \rightarrow \theta_{20}$
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 *TrainningData2* 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 *TrainningData2* 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*.

Table 2
Training algorithm for the naive Bayes classifier using the EM algorithm

Given:
a set UE of unlabeled examples
Initialize the parameter set of *Classifier1* to θ_{10} to label UE.
$\theta_1 \rightarrow \theta_{10}$
Use the labeled examples in UE to estimate the parameter $Pr(c_i \mid l_j)$ and $Pr(l_j)$ of the *classifier2* with $Pr(l_j \mid d) \in \{0, 1\}$.
Loop until the parameters of *classifier2* do not change or the number of iterations exceeds a predefined value:
(E-step) Estimate the probabilistically-weighted class labels, $Pr(l_j \mid d)$, for every document using Eq 12. (M-step) Use the estimated class labels, $Pr(l_j \mid d)$, to calculate new parameters using all documents, by Eqs 10 and 11.

3.1. Expectation-maximization algorithm

EM is a class of iterative algorithm for maximum likelihood or maximum a posteriori estimation in problems with incomplete data [2]. We applied EM to the problem of Web page classification by considering the unlabeled data as the incomplete data. EM consists of two steps, which are the expectation step (E-step) and the maximization step (M-step). The E-step's purpose is to estimate the membership of each unlabeled data by assigning the probability for each class to each unlabeled data. The M-step estimates new classifier parameters using all documents. We combined naive Bayes with EM in the same way as stated in [8]. As shown in Table 2, the first step starts with the parameter estimation of the *classifier1* 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. Our method for training naive Bayes with the EM algorithm is shown below.

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

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

Where T is the total number of unique characters in the training set, $N(c_i, d)$ is the number of times character c_i occurs in document. The probability of a class label is given by Eq. 11:

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

Where $|L|$, $|D|$ are the number of class labels and the number of documents in the training set. Given an unlabeled document d, of n character $(c_1, c_2, \ldots, c_n)$, the naive Bayes classifier estimates the probability that the document belongs to class label l_j by using Eq. 12 below.

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

Note that now $Pr(l_j|d)$ is a probabilistically-weighted value; each document d is considered to be of class label l_j with probability equal to the estimate $Pr(l_j|d)$. The algorithm used the word segmentation classifier once for determining the initial labels for the training data.

In the problem of Thai/non-Thai Web page classification, we employed the word segmentation classifier with an initial value, θ_{10}, for labeling each unlabeled Web page. The experiments were conducted using bag-of-characters representation. For the Web page categorization problems, a small number of initial labeled data were supplied to the naive bayes classifier in order to determine the initial value, θ_{10}, for labeling each unlabeled Web page and bag-of-words representation was used.

3.2. *Co-training algorithm*

The Co-Training algorithm was first introduced by Blum and Mitchell [1]. The concept of the algorithm is based on the boosting technique. That means, the algorithm learns from a small amount of initial labeled data and then incrementally classifies unlabeled data into categories. The basic assumption of Co-Training is that the instance distribution is compatible with the target function. It requires that, for most examples, the target functions over each feature set predict the same label. For example, in the Web page domain, the class of the instance should be identifiable using either the hyperlink text or the page text alone. The second assumption is that the first features 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 algorithm is shown in Table 3.

3.3. *Supervised naive bayes classifier*

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

4. Experimental results

In order to assess the performance of the ICT algorithm, we conducted three experiments on Web page classification problems. The objective of the first experiment is to evaluate the effectiveness of the algorithm when domain knowledge is available. The other experiments involved a more difficult problem to see the performance of ICT when no domain knowledge is given. This section is organized as follows. Section 4.1 gives details about the performance measurements. Section 4.2 is the result for Thai/non-Thai Web page classification. Section 4.3 explains about Web page classification on the WebKb data set. Section 4.4 is the experimental result of the WebClass data set.

4.1. Performance measurements

The effectiveness of an algorithm can be evaluated using the following measurements:

(1) Precision (P)
Precision is a standard measure of the information retrieval (IR) performance, it 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 Eq. 13.

$$\text{Precision} = \frac{\text{no. of correctly predicted positive examples}}{\text{no. of predicted positive examples}} \tag{12}$$

(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 Eq. 14.

$$\text{Recall} = \frac{\text{no. of correctly predicted positive examples}}{\text{no. of all positive examples}} \tag{13}$$

(3) F1-measure (F_1)
The F_1-measure was introduced by van Rijsbergen [13]. It combines precision and recall with equal importance into a single parameter for optimization and is defined as in Eq. 15.

$$F_1 = \frac{2PR}{P+R} \tag{14}$$

where P and R stand for precision and recall, respectively.

4.2. The results on the thai/non-thai web page classification

The objective of this experiment was to assess the effectiveness of ICT in the case that we supplied the knowledge (in the form of dictionary) to a classifier (word segmentation classifier). In the following subsections, we describe the data set and experimental setting for algorithms and the results.

Table 3
The Co-Training algorithm

Given: a set LE of labeled training examples a set UE of unlabeled examples Create a pool UE' of examples by choosing u examples at random from UE. Loop until no examples are 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'.

4.2.1. *Data set and experimental setting*

We collected the data set starting with four Web pages: a Japanese Web page,[3] two Thai Web pages,[4] and an English web page[5] 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 were 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 3) were set in a similar way as in [1]. 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.

4.2.2. *Experimental results*

The results are shown in Table 4. In the table, "Co-Training(Bayes)" and "Co-Training(Word)" were the results of naive Bayes and word segmentation classifiers of Co-Training, respectively. "ICT(Bayes)" and "ICT(Word)" were 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, which was comparable to S-Bayes. The performance of ICT(Bayes) was higher than S-Word. Both classifiers of Co-Training had lower performance, compared to the other classifiers.

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. From the experiments, we

[3] http://www.yahoo.co.jp.
[4] http://www.sanook.com, http://www.pantip.com.
[5] http://www.javasoft.com.

Table 4
The precision (%), recall (%) and F_1-measure of the classifiers for the problem of Thai/non-Thai page classification

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

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. The learning process of ICT took 14.5 second, whereas Co-Training and EM took 51.5 and 37.2 second, respectively. Note that all of the algorithms were implemented using JAVA programming language and were run on Windows platform.

4.3. *The results of web page classification on the webKb data set*

The WebKb data set contains many Web pages related to the university domain. It was obtained via ftp from Carnegie Mellon University [11]. The data set consisted of 981 Web pages collected from the computer science department Web sites at four universities: Cornell, University of Washington, University of Wisconsin, and University of Texas. These Web pages were hand-labeled into four categories, which were course homepages, faculty homepages, project homepages and student homepages.

In this data set, some categories were actually closely related and this made 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.

4.3.1. *Data set and experimental setting*

We had 220 course Web pages, 147 faculty Web pages, 81 project Web pages and 533 student Web pages. Each sample was filtered to remove words that gave no significance in predicting the class of the document. Words to be eliminated were auxiliary verbs, prepositions, pronouns, possessive pronouns, phone numbers, digit sequences, dates and special characters. Then, the word stemming process was applied to each sample by using Porter algorithm [10] in order to remove all suffixes and search for similar words based on the root word. Finally, we extracted all headings appearing in each Web page to be the features of the heading-based classifier. Therefore, each Web page could be viewed as the set of words appearing in the page's content and the set of words appearing in all headings. The first classifier used the words appearing on the headings as the feature set, whereas the second classifier used words appearing in the content of the Web page as the feature set. The combined classifier predicted the class of examples based on the output from the heading-based and content-based classifiers as shown in Eq. 16.

$$Pr(l_j \mid d_i) = Pr(l_j \mid x_1)Pr(l_j \mid x_2) \tag{15}$$

where x_1 and x_2 are the heading and the content feature sets of document d_i. The settings for the classifiers were 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.
(2) For Co-Training and EM, we used the same set of initial labeled data as ICT. The unlabeled data and the test set were also the same. The parameter p and n of Co-Training were set to 1 and 3, respectively.
(3) For the supervised naive Bayes classifier, we supplied the algorithm with 60% of the examples as labeled data and 40% of all samples were used as a test set.

4.3.2. Experimental results

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). After the learning process of every algorithms were accomplished, the performances of the algorithms were evaluated based on the feature sets.

Figures 3–5 show the results of classifiers using the heading feature set, the content feature set and the combined feature set, respectively. In the figures, "ICT" stands for the Iterative Cross-Training algorithm, "S-Bayes" stands for the supervised naive Bayes classifier. Co-Training and EM stands for the Co-Training and Expectation-Maximization algorithm, respectively. Considering the performance measured based on F_1 of classifiers using heading features (Fig. 3), we found that ICT got 80.73% on course homepages which was equal to Co-Training but higher than EM. This fact is also true for the student homepages, ICT got 92.45%, whereas Co-Training and EM got only 85.13% and 89.20%. For the project homepages, ICT obtained 50.06%, where as Co-Training and EM obtained 37.84% and 51.26%. Nevertheless, ICT gave the competitive performance compared to S-Bayes. S-Bayes got 81.13%, 50.00%, 55.56% and 91.16% measured on course homepages, faculty homepages, project homepages and student homepages, respectively.

For the content feature set as shown in Fig. 4, the F_1 score of ICT was 84.47% and was higher than those of Co-Training and EM on student homepages which were 66.27% and 77.05%, respectively. Nevertheless, the performance measured on other categories are less than those of Co-Training, EM and S-Bayes.

Figure 5 shows the performance measured on classifiers using the combined feature set (heading + content). The performance of ICT was higher than those of Co-Training and EM on course homepages and student homepages.

Note that we applied the micro average to measure the overall performance of each classifier. The micro average is normally used when the number of test data in each category are different. The average performance of all classifiers are given both in Fig. 6 and Table 5. 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 S-Bayes. This is because ICT employed only 50% of the labeled data used by S-Bayes. 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 and EM, it took more than 20 minutes to converge (all of the algorithms were implemented using JAVA language on Windows platform).

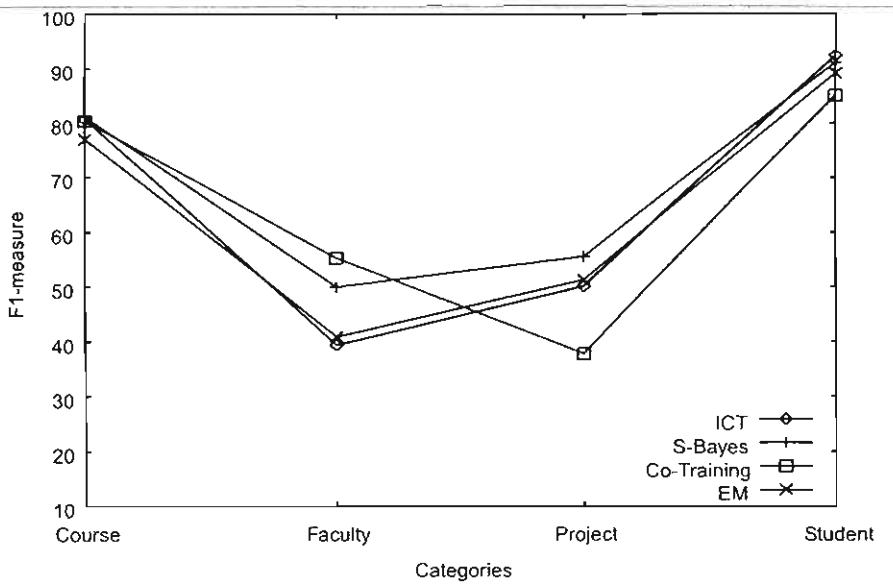


Fig. 3. The performance of classifiers using the heading feature set on the WebKb data set.

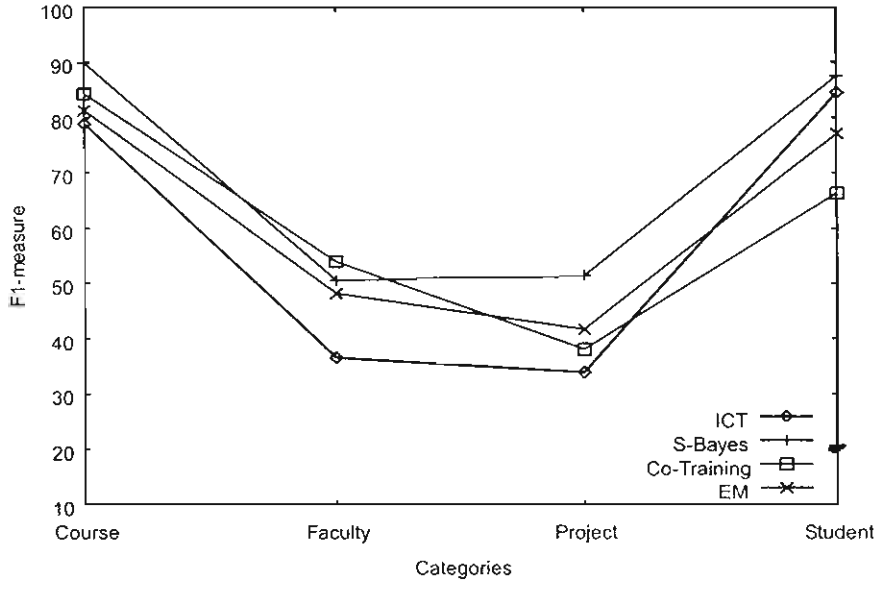


Fig. 4. The performance of classifiers using the content feature set on the WebKb data set.

4.4. *The results of web page classification on the webClass data set*

The WebClass data set was obtained from a machine learning research group in Italy [12]. It consists of 92 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 categories are both concerning about vehicles and are closely related.

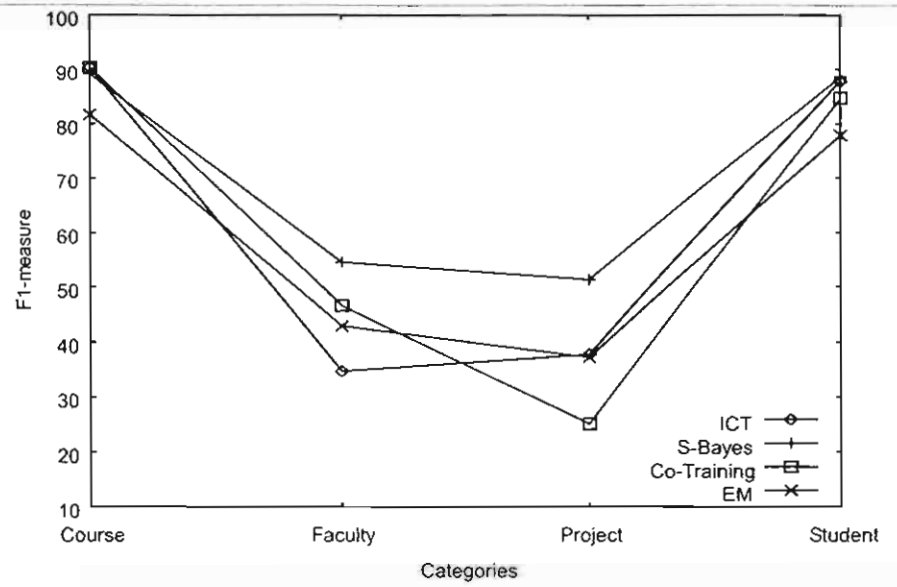


Fig. 5. The performance of classifiers using the combined feature set on the WebKb data set.

Table 5
The average performance of classifiers on the WebKb data set

Classifier	Heading			Content			Heading+Content		
	P (%)	R (%)	F_1	P (%)	R (%)	F_1	P (%)	R (%)	F_1
ICT	71.85	94.39	78.25	67.23	86.73	71.76	73.39	88.27	76.22
S-Bayes	74.99	87.24	79.91	76.95	84.14	79.60	78.92	83.76	80.46
Co-Training	73.95	84.69	75.64	79.69	60.72	66.14	68.29	87.26	75.30
EM	68.62	91.64	75.98	76.78	74.28	70.70	74.87	78.50	70.08

4.4.1. Experimental setting

The preprocessing step was done in the same way as in the WebKb data set. The settings for classifiers were 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.

4.4.2. Experimental results

All experiments were conducted using 3-fold cross-validation. Figures 7–9 show the results of classifiers measured based on heading, content and the combined feature sets.

Considering performance measured based on heading features on classifiers as shown in Fig. 7, ICT got 96.55% on astro homepages which was higher than those of S-Bayes, Co-Training and EM which got 84.08%, 76.47% and 89.51%, respectively. For the motorcycle homepages, ICT got 84.62% which was higher than those of S-Bayes, Co-Training and EM which got 81.27%, 70.04% and 52.72%, respectively. Nevertheless, the performances of the other categories, auto and jazz, of ICT were less than those of S-Bayes, Co-Training and EM.

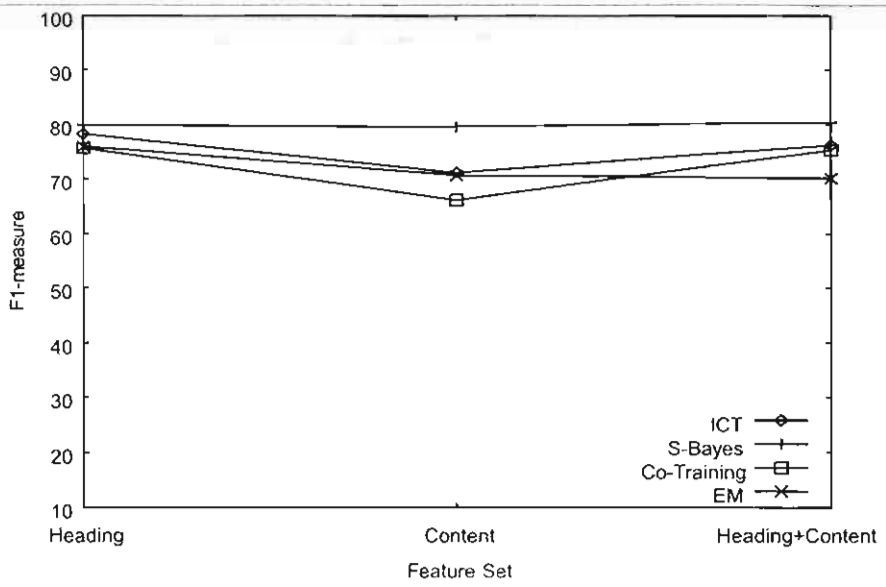


Fig. 6. The average performance of classifiers using different feature sets on the WebKb data set.

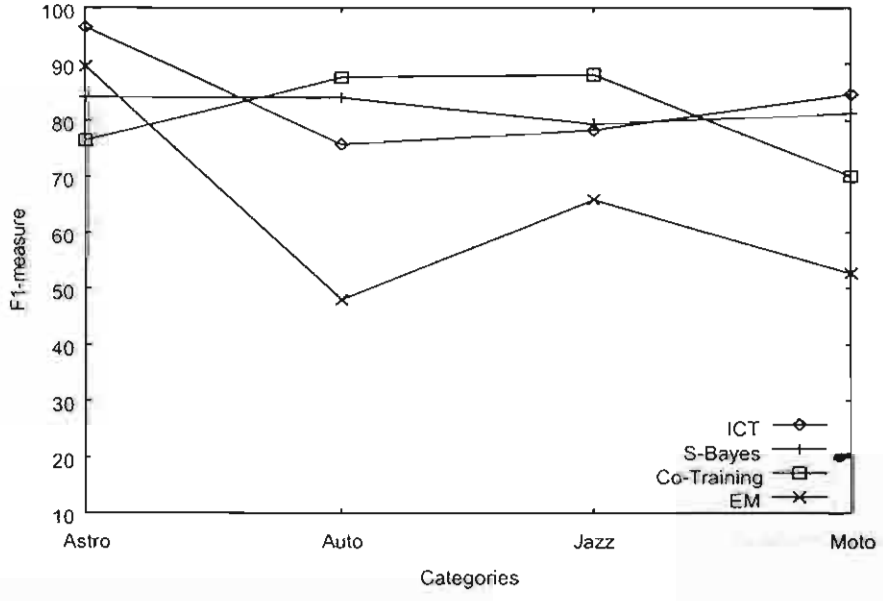


Fig. 7. The performance of classifiers using the heading feature set on the WebClass data set.

Figure 8 shows the performance of classifiers using the content feature set. We found that ICT got higher performance than other classifiers on auto, jazz and motocycle categories. The performance measured based on the combined feature sets is shown in Fig. 9. We found that ICT got higher performance on jazz and motocycle categories (100.00%, 96.55%), whereas S-Bayes, Co-Training and EM got 98.85%, 92.97%, 97.78% on the jazz category and 90.39%, 82.91%, 85.35% on the motorcycle category.

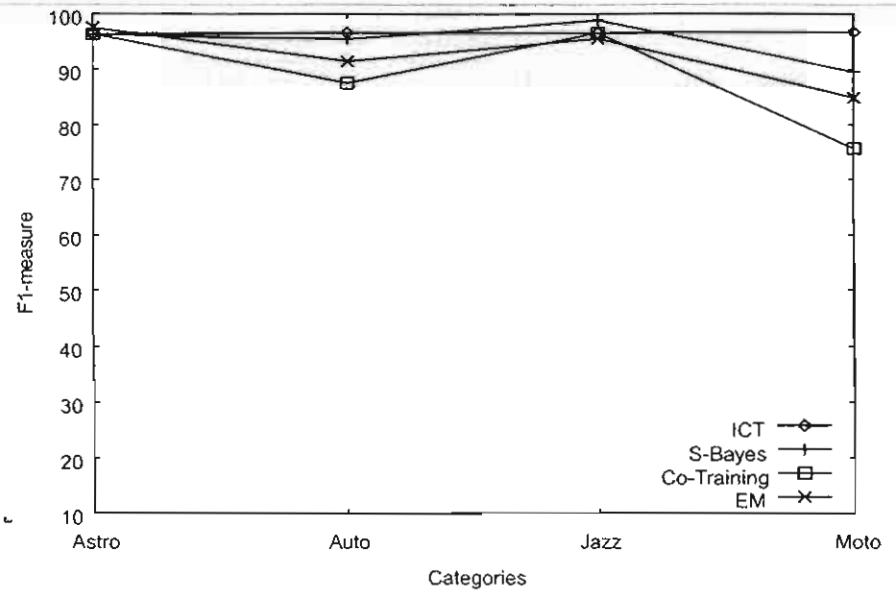


Fig. 8. The performance of classifiers using the content feature set on the WebClass data set.

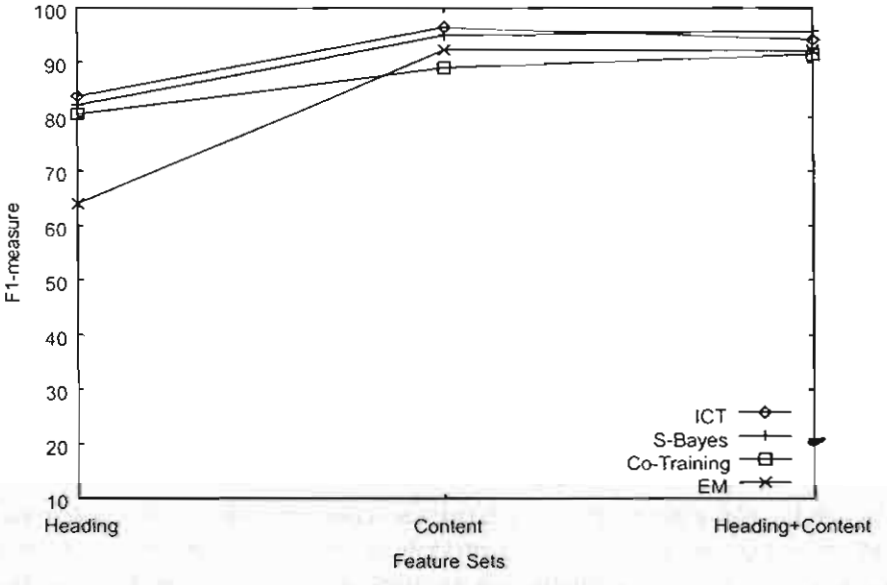


Fig. 9. The performance of classifiers using the combined feature set on the WebClass data set.

Considering the average performance measured by F_1 (as shown in Fig. 10 and Table 6), 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 classifiers using the combined feature set of ICT was higher than those using S-Bayes. However, the performance deficiency of the combined feature set (heading+content-based classifier) was less than those using S-Bayes. For the heading-based classifier, Co-Training and EM lost

Table 6
The average performance of classifiers on the WebClass data set

Classifier	Heading			Content			Heading+Content		
	P (%)	R (%)	F_1	P (%)	R (%)	F_1	P (%)	R (%)	F_1
ICT	86.47	85.72	83.78	95.00	98.22	96.49	92.45	96.43	94.18
S-Bayes	84.36	85.12	82.17	93.14	97.62	94.99	93.97	98.21	95.81
Co-Training	74.16	92.86	80.51	83.00	98.22	89.01	87.62	97.02	91.43
EM	68.11	69.64	64.00	87.60	98.81	92.31	87.31	99.40	92.21

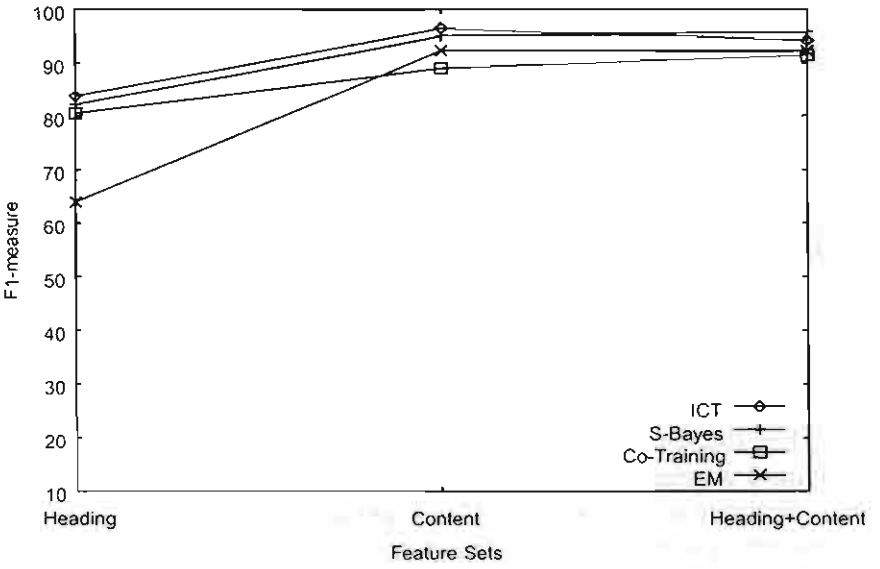


Fig. 10. The average performance of classifiers using different feature sets on the WebClass data set.

2.02% and 22.11%, respectively.

For the content-based classifiers, 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. Note that the training time of ICT is much less than Co-Training and EM. With ICT, it took 1 minute for the algorithm to converge, whereas Co-Training took 5 minutes and EM took 2 minutes.

4.5. *Performance of ICT on noisy 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 a learning algorithm is affected by this real world problem. Since ICT, Co-Training and EM are boosting-style learning algorithms, they need a small amount of initial labeled data. 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. 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 Fig. 11, ICT was robust in the presence of noise as neither classifier's performance changed. 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

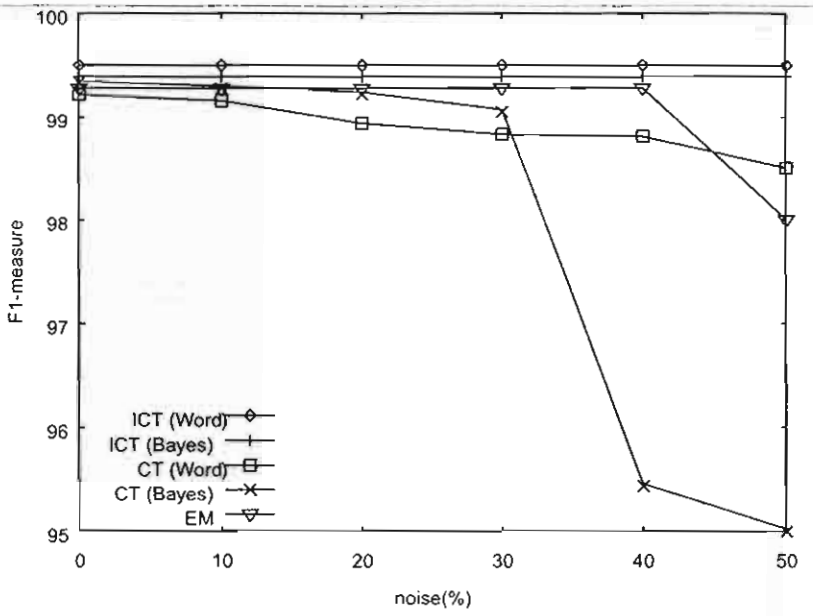


Fig. 11. The performance of classifiers at different levels of noise on the Thai/nonThai data set.

relabeled in every iterations by both classifiers. Both classifiers, CT (Word) and CT (Bayes), of Co-Training 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 Co-Training used an incremental labeling style and thus 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 Co-Training. The performance of the EM algorithm dropped slightly when noise was increased to 50%.

The graphs in Fig. 11 show 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 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 performances of all classifiers when noise was added on the WebClass data set are shown in Fig. 13. 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 24.10%, 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.

5. Conclusions

We have presented an algorithm that effectively uses unlabeled examples to estimate the parameters of the system for classifying Web pages. The method is based on two sub-classifiers that iteratively train

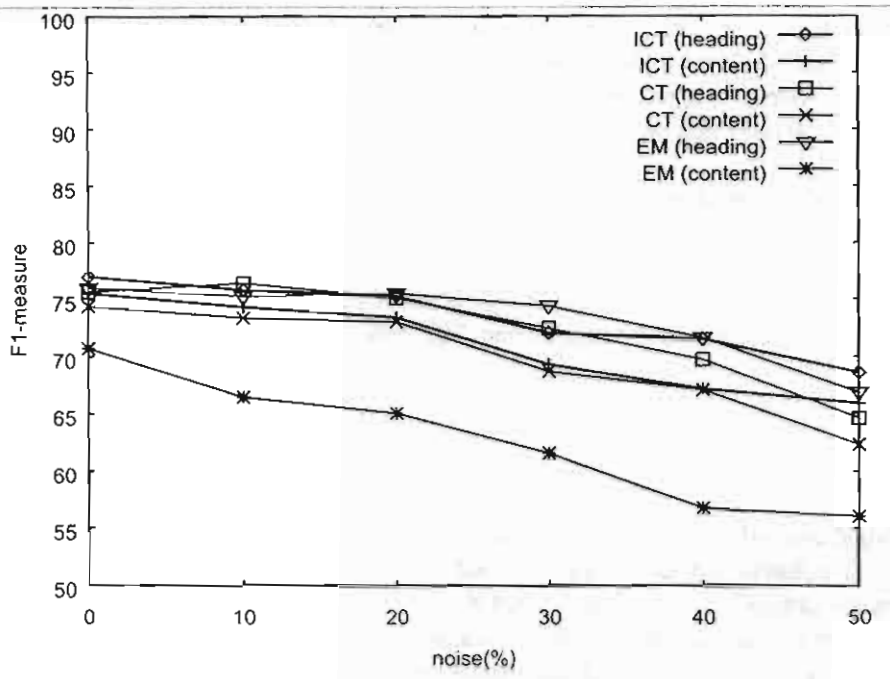


Fig. 12. The performance of classifiers at different levels of noise on the WebKb data set.

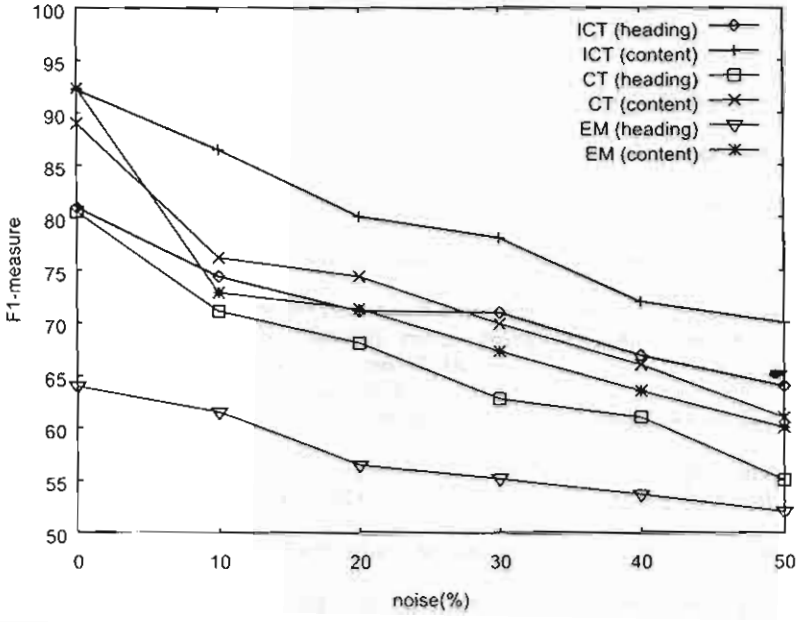


Fig. 13. The performance of classifiers at different levels of noise on the WebClass data set.

each other. Since ICT consists of two sub-classifiers, the algorithm has an ability to utilize different feature sets of the Web pages during the learning process to construct the most appropriate model for classifying unseen Web pages. With no or a small set of pre-labeled examples, our method gives high

precision and recall on classifying Web pages. The performance of our method is comparable with those of supervised ones, which demonstrates the successful use of unlabeled data of our algorithm. We have applied ICT to three classification problems. When we supply domain knowledge to the stronger classifier, ICT has an ability to boost the performance of the weaker classifier. Moreover, ICT's performance is still acceptable, when no domain knowledge is given.

ICT has an advantage in that it quickly converges since the labeling style is re-labeling mode, unlike Co-Training which uses an incremental labeling style. ICT is robust in the presence of noise, when we can provide domain knowledge to the algorithm. ICT is an easy to use algorithm, since we need not tune many parameters, unlike the Co-Training algorithm which requires the parameter tuning of p and n. In the case that no domain knowledge was available, ICT performance declines less than Co-Training and EM.

Acknowledgements

The authors would like to thank Prof. Luc De Raedt from the Machine Learning and Natural Language Processing Research Group, University of Freiburg, for his suggestions and valuable comments. This research has been supported by the Thailand Research Fund, and National Electronics and Computer Technology Center. The views and conclusions contained in this paper are those of the authors and the Thailand Research Fund or National Electronics and Computer Technology Center is not necessarily of the same opinion.

References

[1] A. Blum and T. Mitchell, *Combining labeled and unlabeled data with cotraining*, in Proc. Int. Conf. the 11th Annual Conference on Computational Learning Theory, 1998.
[2] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum likelihood from incomplete data via the em algorithm, *Royal Statistical Society* 39(1) (1997), 38.
[3] T. Joachims, *Text categorization with support vector machines: Learning with many relevant features*, in: Proc. Int. Conf. Machine Learning, 1998, pp. 137–142.
[4] A.R. Jones, A. McCallum, K. Nigam and E. Rilo, *Bootstrapping for text learning tasks*, in Proc. Int. Conf. Artificial Intelligence, 1999, pp. 52–63.
[5] A. McCallum and K. Nigam, *Employing em and pool-based active learning for text classification*, 1998, pp. 350–358.
[6] S. Meknavin, P. Charoenpornsawat and B. Kijsirikul, *Feature-based thai word segmentation*, in Proc. Int. Natural Language Processing Pacific Rim Symposium, 1997, pp. 41–48.
[7] T. Mitchell, Machine Learning, McGraw-Hill, 2 ed., 1979.
[8] K. Nigam, A. McCallum, S. Thrun and T. Mitchell, *Text classification from labeled and unlabeled documents using em*, 2000, pp. 103–134.
[9] J.M. Pierre, Practical issues for automated categorization of web sites, *In Proc. Int. Conf. Semantic Web*, 2000.
[10] M.F. Porter, An algorithm for suffix stripping, PhD thesis, available on-line at http://www.dcs.gla.ac.uk/Keith/ Preface.html., 1980.
[11] Web-Kb Project, http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo51/www/co-training/data/course-cotrain-data.tar.gz. Carnegie Mellon University, USA.
[12] WebClass Project, http://www.di.uniba.it/malerba/software/webclass/webclass.html, University of bari, Italy.
[13] C.J. van Rijsbergen, *Information Retrieval. Dept. of Computer Science*, University of Glasgow, 2 ed., 1979.
[14] Y. Yang and J. Pederson, *Feature selection in statistical learning of text categorization*, In Proc. Int. Conf. Machine Learning, 1997, pp. 412–420.

tapraid5/8c-int/8c-int/8c1203/8c0419-03a | franklim | S=4 | 9/22/03 | 7:18 | Art: 3049L | Input-bsu(bsu)

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

AQ: 1

Nuanwan Soonthornphisaj,* Boonserm Kijsirikul†
Machine Intelligence and Knowledge Discovery Laboratory, Department of Computer Engineering, Chulalongkorn University, Bangkok 10330, Thailand

The article presents a new learning method, called *iterative cross-training* (ICT), for classifying Web pages in three classification problems, i.e., (1) classification of Thai/non-Thai Web pages, (2) classification of course/non-course home pages, and (3) classification of university-related Web pages. Given domain knowledge or a small set of labeled data, our method combines two classifiers that are able to use effectively unlabeled examples to iteratively train each other. We compare ICT against the other learning methods: a supervised word segmentation classifier, a supervised naïve Bayes classifier, and a co–training-style classifier. The experimental results on three classification problems show that ICT gives better performance than those of the other classifiers. One of the advantages of ICT is that it needs only a small set of prelabeled data or no prelabeled data in the case that domain knowledge is available. © 2003 Wiley Periodicals, Inc.

AQ: 2

1. INTRODUCTION

Given prelabeled training data, supervised learning has been applied successfully to text classification.[1,3,4,7,8,10,18] However, one of the difficulties of using supervised learning is that the algorithm needs a large number of labeled examples to find the common properties of the class and use them to classify unseen data. Unfortunately, the text classification is a tedious job and a time-consuming task for humans to read and analyze the category of the pages. Although it is costly to construct hand-labeled data, in some domains it is easy to obtain unlabeled data such as data on the Internet. Thus, if we are able to use effectively the available unlabeled data, we will simplify the task of building text classifiers. Various methods have been proposed to use unlabeled data together with prelabeled data for text classification, such as *active learning with committee,*[11] text classification using EM,[15] and the *co-training* algorithm.[2]

AQ: 3

AQ: 4

*Author to whom all correspondence should be addressed: e-mail: nuanwan@mind.cp.eng.chula.ac.th.
†e-mail: boonserm@mind.cp.eng.chula.ac.th.

AQ: 20

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 18, 000–000 (2003)
 Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.10157

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

Because the Internet becomes a part of our life, everyone could access any Web pages through a search engine. To provide the high impact to users and apply our learning algorithm to the real-world application, we intend to apply our learning method in the area of the Internet that focused on Web page classification. Therefore, we apply our algorithm to three classification problems: (1) the classification of Web pages based on the specific language (Web pages written in Thai language and non-Thai language); (2) the classification of Web pages into course and noncourse home pages, which was introduced by Blum and Mitchell[2]; and (3) the classification of the university-related home page, which was introduced by Craven et al.[5] To evaluate the effectiveness of our method, we also implement other classifiers to compare empirically with our method. The implementation is designed to explain or at least give some answers to questions: Is ICT that combines two classifiers an effective method? Does this kind of combination of two classifiers perform better than only one? Can the method successfully use unlabeled data? The other classifiers are (1) a supervised word segmentation classifier (S-Word), (2) a supervised naïve Bayes classifier (S-Bayes), (3) a co–training-style classifier (CoTraining). Among these classifiers, S-Bayes or S-Word is a single and supervised classifier. CoTraining and ICT are composed of two subclassifiers and are able to use unlabeled data.

The experimental results show that ICT successfully and efficiently classifies Web pages with high precision and recall. The overall performance, evaluated by F_1 measure of ICT is better than those of the other methods tested in our experiments. The better performance of ICT than those of supervised ones (e.g., S-Bayes) shows the successful use of unlabeled data. The results also show that the training technique of ICT also is an effective way because its performance is better than that of CoTraining, which uses a different training technique.

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

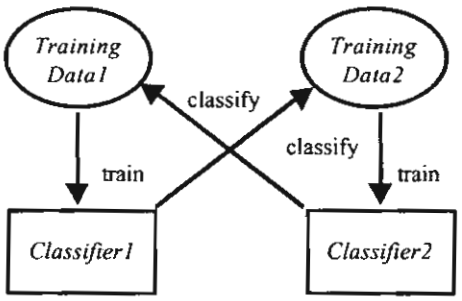


Figure 1. The architecture of ICT is shown. It is composed of two classifiers that use unlabeled data to iteratively train each other.

2. ICT

This section presents ICT. First, we describe the architecture of our learning system and then give the details of two classifiers used in the system. Figure 1 F1 shows our learning system, which determines how to classify Web pages. The system is composed of two classifiers: *Classifier1* and *Classifier2*. Given domain knowledge or a small set of prelabeled data, these two classifiers estimate their parameters from unlabeled data by receiving training from each other. Two training data sets, called *TrainingData1* and *TrainingData2* are duplicated from the unlabeled data provided by the user. Let θ_1 and θ_2 be sets of parameters of *Classifier1* and *Classifier2*, respectively. *TrainingData1* is used to train *Classifier1* to estimate its parameter set, and *TrainingData2* is used to estimate the parameter set of *Classifier2*. The algorithm for training the classifiers is shown in Table I. T1

The idea behind our algorithm is that if we can obtain reliable statistical information contained in *TrainingData2*, it should be useful in classifying *TrainingData1*. If the starting parameter set of *Classifier1* (θ_{10}) has the property that it produces more true positive than wrong positive examples and more true negative than wrong negative examples for *TrainingData2*, the statistical information in correctly classified examples will be obtained.

Using this information, *Classifier2* should correctly classify more examples in *TrainingData1* that have similar characteristics. If the newly labeled *TrainingData1* can produce θ_1 better than θ_{10}, more reliable parameters of the whole system should be obtained after each iteration.

In the algorithm, first, we initialize the parameter sets of *Classifier1* and *Classifier2*. This is done by training the classifiers with a small set of labeled examples if they are available. If no labeled example is provided to the system, the values of the parameters can be a predetermined or randomly chosen. When a classifier labels data, it can ask for the confirmation from the other classifier to make decisions about which class the example should be. If both classifiers agree with the same classifying result, that example will be labeled. The purpose of consistency checking is to produce more reliable labeled data, but the checking will slow down the learning process.

Table I. The training algorithm of ICT.

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

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

Train *Classifier2* by using labeled examples in *TrainingData2* to estimate the parameter set θ_2 of *Classifier2*.
If *labeling_mode* = BATCH, Then
- Use *Classifier2* with the current parameter set θ_2 to label all data in *TrainingData1* into positive examples and negative examples, and check consistency of the classification with *Classifier1* if necessary.

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

Train *Classifier1* by the labeled examples in *TrainingData1* to estimate the parameter set θ_1 of *Classifier1*.

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

The following sections describe the details of the classifiers.

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

In the problem of classification of Thai/Non-Thai Web pages, our goal was to classify Web pages into Thai and non-Thai pages. This problem is of interest because we want to build a Web robot that efficiently crawls the Web and retrieves only Thai pages for building a Thai search engine. In this problem, the first subclassifier *Classifier1* is given some knowledge about the domain in the form of a dictionary and uses the dictionary for helping in determining whether a page is

written in Thai or not. The algorithm used by *Classifier1* is the word segmentation algorithm, which will be described in the next section. The second subclassifier *Classifier2* is given no knowledge and uses the naïve Bayes classifier.

2.1.1. *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 can not expect that all words in the page appear in the dictionary because the Web page usually contains names of persons, organizations, etc. not occurring in the dictionary and may contains words written in foreign languages. Therefore, it is necessary to determine how many words should be contained. This task is more difficult when it is considered in a language that has no word boundary delimiters such as Thai, Japanese, etc.[13]

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

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

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

where $\operatorname{cost}(w_i)$ is $\eta 1$ if w_i is a word in the dictionary and $\eta 2$ 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. Because generating all possible segmentations and calculating their costs is very expensive, we use a dynamic programming technique to implement this calculation. Note that any sequence of characters, $c_i, \ldots, c_j$, found in the dictionary must be considered as a word and must not be grouped with nearby characters to form a long unknown string.

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

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

Given sets of positive and negative examples, the classifier finds the threshold of *WordRatio* that maximizes the number of correctly classified positive and negative examples. If *WordRatio* of a page is greater than the threshold, we will classify it as positive (Thai page). Otherwise, we will classify it as negative (non-Thai page).

For simplicity, let us use only the threshold of *WordRatio* as the parameter of the word segmentation classifier (θ_1).

Having only the threshold of *WordRatio* (θ_1) as the parameter, we can find θ_{10}, which produces more true positive and true negative examples for *TrainingData2*. As described previously, most of Thai pages should have a high value of *WordRatio*, whereas non-Thai pages should have a low value of *WordRatio*. If the numbers of Thai and non-Thai pages in *TrainingData2* are the same, it is easy to see that any value of θ_{10} will give more correctly classified pages than incorrectly classified pages (except for $\theta_{10} = 0.0$ or $\theta_{10} = 1.0$, which gives the same number of correctly and incorrectly classified pages). If the number of Thai pages is lower than the number of non-Thai pages, a high value of θ_{10} (e.g., 0.7, 0.8, and 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 used when the number of Thai pages is larger than that of non-Thai pages. AQ: 5

A new θ_1 can be estimated after the naïve Bayes classifier (*Classifier2*) labels data in *TrainingData1*. Let *SP* be the smallest value of *WordRatios* from all labeled positive examples, and let *LN* be the largest value from all labeled negative examples. In case of $SP \geq LN$, the new θ_1 is estimated as

$$\theta_1 = \frac{SP + LN}{2} \tag{2}$$

Now, consider the case of $SP < LN$. Let $V_1 = SP$, $Vn = LN$, and $V_2, \ldots, V_{n-1}$ be the values between V_1 and V_n ($V_1 \leq V_2 \leq \cdots \leq V_{n-1} \leq V_n$). The new θ_1 is estimated as

$$\theta_1 = \frac{V_{i^*} + V_{i^*+1}}{2}$$

$$V_{i^*} = \underset{V_i}{\operatorname{argmin}} (\text{no. of } V_j + \text{no. of } V_k) \tag{3}$$

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 $SP > LN$, θ_1 will completely discriminate the labeled positive from negative examples. Otherwise, θ_1 will give the minimum errors of misclassified examples.

2.1.2. *Naïve Bayes Classifier (Classifier2)*

For text classification, naïve Bayes is among the most commonly used and the most effective methods.[14] To represent text, the method usually uses *bag-of-words* representation. Instead of bag-of-words, we use the simpler *bag-of-characters* representation in the problem of classification of Thai/non-Thai pages. This representation is suitable for a Web robot to identify Thai Web pages because it requires no word segmentation and thus it is very fast. In spite of its simplicity, our results show the effectiveness of bag-of-characters representation in identifying Thai Web pages, as shown later in Section 4.

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

$$l^* = \underset{l_j}{\operatorname{argmax}} \Pr(l_j|c_i, \ldots, c_n)$$

$$= \underset{l_j}{\operatorname{argmax}} \frac{\Pr(l_j)\Pr(c_1, \ldots, c_n/l_j)}{\Pr(c_1, \ldots, c_n)} \tag{4}$$

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

In our case, L is the set of positive and negative class labels. The term $\Pr(c_1, \ldots, c_n)$ in Equation 4 can be ignored, because we are interested in finding the most likely class label.

Because there are usually an extremely large number of possible values for $d = (c_1, c_2, \ldots, c_n)$, calculating the term $\Pr(c_1, c_2, \ldots, c_n|l_j)$ requires a huge number of examples to obtain reliable estimation. Therefore, to reduce the number of required examples and improve reliability of the estimation, assumptions of naïve Bayes are made.[14] These assumptions are (1) the conditional independent assumption, i.e., the presence of each character is conditionally independent of all other characters in the document given the class label and (2) an assumption that the position of a character is unimportant, e.g., encountering the character "a" at the beginning of a document is the same as encountering it at the end. Clearly, these assumptions are violated in real-world data, but empirically naïve Bayes has been applied successfully in various text classification problems.[8,12,19]

Using the foregoing assumptions, Equation 5 can be rewritten as

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

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

This model also is called the *unigram* model because it is based on statistics about a single character in isolation.

The probabilities $\Pr(l_j)$ and $\Pr(c_i|l_j)$ are used as the parameter set θ_2 of our naïve 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:

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

where $N(c_i, l_j)$ is the number of times character c_i appears in the training set from class label l, $N(l_j)$ is the total number of characters in the training set for class label

l, and T is the total number of unique characters in the training set. Equation 7 uses Laplace smoothing (adding one to all the character counts for a class) to avoid assigning probability values of zero to characters that do not occur in the training data for a particular class.

2.2. Subclassifiers in ICT for the Classification of Course/Noncourse Home Pages

The problem of classification of Web pages into course/noncourse home pages is described in Ref. 2. The course home page gives information about the subjects taught at the university, course syllabus, instructor, class schedule, etc., and noncourse home pages are instructor home pages, department home pages, and student home pages. In this problem, each Web page contains two sets of features: (1) words appearing in the content of the page and (2) words appearing on the hyperlinks that link to that page. Therefore, each page can be viewed in two different ways, i.e., content-based features and hyperlink-based features. With these two feature sets, we construct two naïve Bayes classifiers: the first one (*Classifier1* in Table I) learns from hyperlink features and the second one (*Classifier2*) learns from content-based features. Both classifiers use the naïve Bayes algorithm, which is the same algorithm described in Section 3.1, except that for this problem it uses *bag-of-word* representation.

2.3. Subclassifiers in ICT for the Classification of University-Related Web Pages

The goal of this classification problem is to classify Web pages into four categories, which are course, faculty, project, and student Web pages. In this problem, each Web page contains two sets of features: (1) words appearing in the content of the page and (2) words appearing in headings indicated by the hypertext markup language (HTML) tags (e.g., <H1>, <H2>, or <H3>). Therefore, each page can be viewed in two different ways, i.e., content-based features and heading-based features. With these two feature sets, we construct two naïve Bayes classifiers; the first one learns from content-based features and the second one (Classifier 2) learns from heading-based features. Both classifiers use the naïve Bayes algorithm, which is the same algorithm described in the Section 3.1. This problem also uses *bag-of-word* representation.

3. OTHER CLASSIFIERS USED IN COMPARISON

In our experiment, we will compare ICT with the following classifiers:

(1) A supervised word segmentation classifier
(2) A supervised naïve Bayes classifier
(3) A co–training-style classifier

Table II. The co–training-style algorithm.

Given:
- A set LE of labeled training examples
- A set UE of unlabeled examples

Create a pool UE' of examples by choosing u examples at random from UE.
Loop until no examples left in UE:
  Use LE to estimate the parameter set θ_1 of *Classifier1*
  Use LE to estimate the parameter set θ_2 of *Classifier2*
  Allow *Classifier1* with θ_1 to label p-positive and n-negative examples from UE'
  Allow *Classifier2* with θ_2 to label p-positive and n-negative examples from UE'
  Add these self-labeled examples to LE
  Randomly choose $2p + 2n$ examples from UE to replenish UE'

Supervised word segmentation and supervised naïve Bayes classifiers used in our comparison are the same as those described in Section 3.1, except that they are trained by a large number of hand-labeled data. The Co–training-style classifier is described in the next section.

3.1. Co–Training-Style Classifier

The cotraining algorithm is described in Ref. 2. The idea of the algorithm is that an example can be considered in two different views. For example, a Web page can be partitioned into the words occurring on that page and the words occurring in hyperlinks that point to that page.[2] Either view of the example is assumed to be sufficient for learning. The algorithm consists of two subclassifiers, each of which learns its parameter sets from each view of the example.

Based on this idea, we construct a co–training-style algorithm for our problems. The algorithm is shown in Table II. The algorithm uses two subclassifiers: T2
Classifier1 and *Classifier2*. These two classifiers are the same as ones of ICT:

(1) In the case of classification of Thai/non-Thai pages, we view each Web page as a set of words occurring on that page and a set of characters occurring on the page. The word segmentation classifier (*Classifier1*) is used to learn from the view of the word representation, and the naïve Bayes classifier (*Classifier2*) is used for the character representation. The parameters θ_1 and θ_2 of *Classifier1* and *Classifier2* are estimated in the same way as described in Section 2.1.
(2) In the case of classification of course/noncourse home pages, we view each Web page as words occurring on that page and the words occurring in hyperlinks that point to that page. The page-based classifier *Classifier1* learns from words occurring on that page. The hyperlink-based classifier *Classifier2* learns from words occurring in the hyperlinks. For this problem, both *Classifier1* and *Classifier2* are naïve Bayes classifiers.
(3) In the case of classification of university-related Web pages, we view each Web page as words occurring on that page and words appearing in all headings of that page. The content-based classifier *Classifier1* learns from words occurring on that page. The heading-based classifier *Classifier2* learns from words occurring in all headings of the page.

Our co–training-style algorithm is slightly different from the original one in that our algorithm will consume all data in UE. This is done to provide a fair comparison with the other methods.

4. EXPERIMENTAL RESULTS

We conducted experiments to compare ICT with the other classifiers described in the previous section: supervised word segmentation classifier (S-Word), supervised naïve Bayes classifier (S-Bayes), and co–training-style classifier (CoTraining). This section describes the data set, the setting for each classifier, and the results of the comparison on three classification problems: (1) Thai/non-Thai page, (2) course/noncourse home page, and (3) university-related Web page classification.

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

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

4.1.1. Data Set and Experimental Setting

We collected the data set by starting from four Web pages: a Japanese Web page,[a] two Thai Web pages,[b] and an English web page.[c] 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 have ~900 Thai pages because Thai pages may link to ones that are in English or other languages. We also have ~450 Japanese and 450 English pages. All of these pages were divided into three sets, denoted as A, B, and C, each of which contains 600 pages (about 300 Thai pages, 150 Japanese pages, and 150 English pages). Note that HTML markup tags were removed before the training and testing process. We used threefold cross-validation in all experiments for averaging the results.

The settings for the classifiers are as follows:

(1) For ICT, we ran the algorithm with both incremental and batch modes. We refer to incremental-mode ICT and batch-mode ICT as I-ICT and B-ICT, respectively. We used consistency checking for I-ICT and no consistency checking for B-ICT. No label data were given to B-ICT. The initial θ_{10} was set to 0.7. For I-ICT; we gave 18 hand-labeled pages as initial labeled data for the naïve Bayes classifier.

(2) For CoTraining, the values of the parameters of the classifier (in Table II) were set in a similar way as in Ref. 2. Because CoTraining requires a small set of correctly preclassified 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.

[a]http://www.yahoo.co.jp.
[b]http://www.sanook.com, http://www.pantip.com.

Table III. The precision (%), recall (%), and F_1 measure of the classifiers for the problem of Thai/non-Thai page classification.

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

4.1.2. Results

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

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

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

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

The results are shown in Table III. In the table, "CoTraining(Bayes)" and "Co-Training(Word)" are the results of naïve Bayes and word segmentation classifiers of *CoTraining*, respectively. "B-ICT(Bayes)" and "B-ICT(Word)" are for naïve Bayes and word segmentation classifiers of ICT with the batch mode and "I-ICT(Bayes)" and "I-ICT(Word)" are those of the incremental mode. T3

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

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

[d]The F_1 measure has been introduced by van Rijsbergen[17] to combine recall and precision with an equal weight.

4.2. The Results on the Course/Noncourse Home Page Classification

Now, we describe the data set and experimental setting and the results on the course/noncourse page classification problem.

4.2.1. Data Set and Experimental Setting

The data for this experiment is obtained via file transfer protocol (FTP) from Carnegie Mellon University.[c] It consists of 1051 Web pages collected from the computer science department Web sites at four universities: Cornell, University of Washington, University of Wisconsin, and University of Texas. These Web pages have been hand labeled into two categories. We consider the category "course home page" as the positive class and the other as the negative class. In this data set, 22% of the Web pages are course home pages. AQ: 6

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

The settings for the classifiers are as follows:

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

(2) For CoTraining, the values of the parameters of the classifier (in Table II) were set in the same way as in Ref. 2. Because CoTraining requires a small set of preclassified training data, we gave the algorithm with three positive and nine negative examples. In our experiment, we set the values of $|UE|$, p, n, and u to 776, 1, 3, and 75, respectively.

4.2.2. Results

The experimental results are shown in Table IV. In Table IV, I-ICT(Content) and I-ICT(Hyperlink) stand for the page-based and hyperlink-based naïve Bayes classifiers of I-ICT, respectively, and B-ICT(Content) and B-ICT(Hyperlink) are those of B-ICT. CoTraining(Content) and CoTraining(Hyperlink) are page-based and hyperlink-based naïve Bayes classifiers of CoTraining algorithm, respectively. S-Bayes(Content) and S-Bayes(Hyperlink) are supervised naïve Bayes classifiers, T4

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

Table IV. The precision (%), recall (%), and F_1 measure of the classifiers for the problem of course/noncourse page classification.

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

which classify Web pages based on words in Web pages and words in hyperlinks, respectively.

As shown in the Table IV, I-ICT(Content) gave the best performance followed by S-Bayes(Content), S-Bayes(hyperlink), I-ICT(Hyperlink), CoTraining(Hyperlink), and CoTraining(Content). The performance of B-ICT was lower than the others. Compared with the performance of B-ICT on Section 5.1, the results of B-ICT on this problem were not good. This is because of the fact that unlike B-ICT in Section 5.1, which was given knowledge in the form of the dictionary, B-ICT on this problem had no knowledge about the domain. In this problem, B-ICT received only a small set of labeled examples for building its initial parameter set. As shown by the results, this initial parameter set did not contain enough statistical information for labeling the whole examples in batch mode. However, when we ran the algorithm with incremental mode, with the help of consistency checking, I-ICT incrementally added a small set of examples on each round and gave improved results over B-ICT.

The reason that I-ICT(Content) gave better performance compared with S-Bayes is because I-ICT(Content) cooperated with I-ICT(Hyperlink) while S-Bayes used the single classifier. The performance of I-ICT(Hyperlink) was not as good as that of I-ICT(Content). This is because hyperlinks contain fewer words or sometimes it contains only a proper noun. Therefore, it is less capable of building an accurate classifier. The training technique of I-ICT also is an effective way because its performance was better than that of CoTraining, which uses a different training technique.

4.3. The Results on the University-Related Web Pages

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

4.3.1. Data Set and Experimental Setting

This experiment used the WebKB data set,[5] which contains Web pages gathered from the department of computer science in four universities. These Web

pages have been hand labeled into four categories, which are course home page, faculty home page, project home page, and student home page.

In this data set, some categories are actually closely related, which make the classification more difficult. A course home page gives information about the subject such as the course outline, the class schedule, and reference books. A faculty home page is an instructor home page, which provides information about the instructor's research in the teaching course. A project home page actually is a research home page. A student home page is a personal home page of a student who studies at the university. AQ: 7

We have 220 course Web pages, 147 faculty Web pages, 81 project Web pages, and 533 student Web pages. Each example was filtered to remove words that give no significance in predicting to the class of the document as in Section 4.2. Then, the word stemming process was applied to each sample by using Porter the algorithm[16] in order to remove all suffixes and search for similar words based on the root word. Finally, we extracted all headings appearing in each Web page to be the feature of heading-based classifier. Therefore, each Web page can be viewed as the series of words appearing in the page's content and words appearing in all headings belong to the Web page.

For each category, 5% of all examples were used as initial labeled data. The training set was composed of 70% of all examples and 25% of all examples were used as the test set.

We assume that each Web page may belong to more than one category; therefore, the learning process was performed by class basis to get the knowledge for each class. For example, we train the classifier to learn the concept of the course Web page by considering the course Web page as positive examples, whereas examples from other classes are considered to be negative ones. We conducted experiments by using threefold cross-validation for averaging the results of each category. The parameters p and n of the CoTraining algorithm and ICT are set to 1 and 3, respectively.

4.3.2. Results

Tables V and VI show the results of all classifiers using the heading-based feature and the content-based feature, respectively. In the tables, S-Bayes stands for the supervised naïve Bayes classifier, I-ICT and B-ICT are the naïve Bayes classifiers of the ICT algorithm using incremental and batch modes, respectively. CoTraining is the result of the naïve Bayes classifier of the CoTraining classifier. T5-6

Considering the performance of classifiers in Table V, we found that I-ICT outperformed S-Bayes and CoTraining in all categories. The summary result in Table VI shows the explicit comparison of each classifier's performance. According to the F_1 measure, I-ICT(Heading) obtained the best performance with 73.83% followed by S-Bayes(Heading) and I-ICT(Content). Most classifiers using the heading feature obtained higher performance than classifiers using the content feature. It implies that the heading feature has more potential than the content feature in helping the classifier to build the correct model used in the classification process. Considering the feature set, we found that I-ICT obtained the highest

Table V. The precision (%), recall (%), and F_1 measure of classifiers using threefold cross-validation.

Category	Classifier	P (%)	R (%)	F_1
Course	I-ICT(Heading)	87.15	93.94	90.42
	S-Bayes(Heading)	88.51	84.24	86.32
	CoTraining(Heading)	87.91	82.42	85.08
	I-ICT(Content)	90.56	79.40	84.61
	S-Bayes(Content)	94.33	58.79	72.44
	CoTraining(Content)	82.54	49.69	62.03
	B-ICT(Content)	25.00	100.00	40.00
	B-ICT(Heading)	25.00	100.00	40.00
Faculty	I-ICT(Heading)	77.96	66.67	71.87
	CoTraining(Heading)	83.88	56.76	67.71
	S-Bayes(Heading)	77.43	43.24	55.49
	I-ICT(Content)	69.58	34.23	45.89
	B-ICT(Content)	25.00	100.00	40.00
	B-ICT(Heading)	25.00	100.00	40.00
	CoTraining(Content)	68.78	25.23	36.92
	S-Bayes(Content)	55.56	13.51	21.73
Project	I-ICT(Heading)	66.10	73.27	69.50
	S-Bayes(Heading)	87.41	45.15	59.54
	I-ICT(Content)	56.82	47.60	51.80
	B-ICT(Content)	25.00	100.00	40.00
	B-ICT(Heading)	25.00	100.00	40.00
	CoTraining(Content)	53.21	24.22	33.29
	S-Bayes(Content)	93.33	14.80	25.55
	CoTraining(Heading)	49.70	15.07	23.13
Student	I-ICT(Heading)	87.51	41.28	56.10
	B-ICT(Content)	25.00	100.00	40.00
	B-ICT(Heading)	25.00	100.00	40.00
	CoTraining(Heading)	91.67	21.46	34.78
	S-Bayes(Heading)	97.92	19.09	31.95
	I-ICT(Content)	100.00	15.39	26.67
	CoTraining(Content)	80.95	12.30	21.36
	S-Bayes(Content)	100.00	2.50	4.88

Table VI. The precision (%), recall (%), and F_1 measure on the average performance of classifiers using threefold cross-validation.

Classifier	P (%)	R (%)	F_1
I-ICT(Heading)	79.68	68.79	73.83
S-Bayes(Heading)	87.81	47.93	62.01
I-ICT(Content)	79.24	44.16	56.71
CoTraining(Heading)	78.29	43.93	56.28
CoTraining(Content)	71.37	27.86	40.07
B-ICT(Heading)	25.00	100.00	40.00
B-ICT(Content)	25.00	100.00	40.00
S-Bayes(Content)	85.81	22.40	35.53

tapraid5/8c-int/8c-int/8c1203/8c0419-03a | franklim | S=4 | 9/22/03 | 7:18 | Art: 3049L | Input-bsu(bsu)

performance in both features, I-ICT also outperformed S-Bayes because I-ICT combines two classifiers based on different feature sets. For this classification problem, batch-mode ICT did not perform well because it has no domain knowledge, whereas incremental mode of ICT using different labeling technique obtained higher performance than other classifiers.

5. DISCUSSION AND RELATED WORK

We have applied ICT on three classification problems. The problem of Thai/non-Thai page classification is simpler than the problems of course/noncourse home page classification and university-related Web page classification. This can be seen by the performance of all classifiers, which decrease on the second and third problems. For a difficult problem, incremental-mode ICT seems to be more suitable than batch-mode ICT. Batch-mode ICT has an advantage because it runs fast and it is suitable for the problem where we can provide domain knowledge.

Although the performance of our method is comparable or better than the other classifiers, the precision and recall on the problem of course/noncourse and university-related Web page classification are still not high. This may be because of the simple model of the classifiers, i.e., naïve Bayes classifiers. We plan to construct some domain knowledge for giving to the classifier and uses more powerful classifiers to test in this problem in the near future.

Our technique is related to the expectation-maximization algorithm.[6] The EM algorithm is an effective method for dealing with missing values in data and has been applied successfully to text classification.[15] Nigam et al.[15] have shown that the accuracy of classifiers can be improved by using EM to augment a small number of labeled data with a large set of unlabeled data.

Meta-bootstrapping is another unsupervised algorithm for learning from unlabeled data.[9] Like our method, the algorithm is composed of two sublearning algorithms. However, the training process of meta-bootstrapping and the way of using data are different from our method. This algorithm is a multilevel algorithm and is very useful, especially in the complex domain where sublearning algorithms alone could not produce enough good results. We also plan to study this kind of multilevel algorithm for using with our method.

6. CONCLUSION

We have presented a method that effectively uses unlabeled examples to estimate the parameters of the system for classifying Web pages. The method is based on two subclassifiers that iteratively train each other. Because ICT consists of two subclassifiers, the algorithm has an ability to use the different feature sets of the Web pages during the learning process to construct the most appropriate model used in classifying unseen Web pages. Moreover, the ICT algorithm can be applied to other classification problem as well. With no prelabeled or a small set of prelabeled examples, our method gives high precision and recall on classifying Web pages. The performance of our method is competitive with those of supervised ones, which establishes the successful use of unlabeled data of our method.

Acknowledgments

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

References

1. Apte C, Damerau F. Automated learning of decision rules for text categorization. ACM TOIS 1994;12(2):233–251.
2. Blum A, Mitchell T. Combining labeled and unlabeled data with co-training. In: Proc of the 11th Annual Conf on Computational Learning Theory, 1998. AQ: 8
3. Cohen WW. Fast effective rule induction. In: Proc of 12th Int Conf on Machine Learning. San Mateo, CA: Morgan Kaufmann; 1995. AQ: 9
4. Cohen WW, Singer Y. Context-sensitive learning methods for text categorization. ACM Trans Infn Syst 1998;17(2):141–173.
5. Craven M, DiPasquo D, Freitag D, McCallum A, Mitchell T, Nigam K, Slattery S. Learning to extract symbolic knowledge from the World Wide Web. In: AAAI-98, 1998. AQ: 10
6. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc B 1977;39(1):1–38.
7. Joachims J. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In: Proc of the 14th Int Conf on Machine Learning. San Mateo, CA: Morgan Kaufmann; 1997. pp 143–151. AQ: 11
8. Joachims T. Text categorization with support vector machines: Learning with many relevant features. In: Proc of the 10th Eur Conf on Machine Learning. New York: Springer Verlag; 1998. AQ: 12
9. Jones R, McCallum A, Nigam K, Riloff E. Bootstrapping for text learning tasks. IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications. 1999. pp 52–63. AQ: 13
10. Lewis D. Naive (Bayes) at forty: The independence assumption in information retrieval. In: Proc of the 10th Eur Conf on Machine Learning. 1998. AQ: 14
11. Liere R, Tadepalli P. Active learning with committees for text categorization. In: Proc of the 14th Natl Conf on Artificial Intelligence. 1997. pp 591–596. AQ: 13
12. McCallum A, Rosenfeld R, Mitchell T, Nigam A. Improving text classification by shrinkage in a hierarchy of classes. Proc of the 15th Int Conf on Machine Learning. San Mateo, CA: Morgan Kaufmann; 1998. pp 350–358. AQ: 15
13. Meknavin S, Charoenpornsawat P, Kijsirikul B. Feature-based Thai word segmentation. In: Proc of Natural Language Processing Pacific Rim Symposium '97, 1997. AQ: 14
14. Mitchell T. Machine learning. New York: McGraw-Hill; 1997. pp 180–184.
15. Nigam K, McCallum A, Thrun S, Mitchell T. Text classification from labeled and unlabeled documents using EM. Mach Learning 2000. In press. AQ: 16
16. Porter MF. An algorithm for suffix stripping. 1980. pp 130–137. AQ: 17
17. van Rijsbergen CJ. Information retrieval. London: Butterworths; 1979.
18. Yang Y. An evaluation of statistical approaches to text categorizing. Inf Retriev J 1999. AQ: 18
19. Yang Y, Pederson J. Feature selection in statistical learning of text categorization. In: Proc of the 14th Int Conf on Machine Learning. San Mateo, CA: Morgan Kaufmann; 1997. pp 412–420. AQ: 19

THE EFFECTS OF DIFFERENT FEATURE SETS ON THE WEB PAGE CATEGORIZATION PROBLEM USING THE ITERATIVE CROSS-TRAINING ALGORITHM

Nuanwan Soonthornphisaj and Boonserm Kijsirikul
Machine Intelligence & Knowledge Discovery Laboratory
Department of Computer Engineering
Chulalongkorn University,
Phathumwan, Bangkok, 10330, Thailand.
Email : nuanwan@mind.cp.eng.chula.ac.th , boonserm@mind.cp.eng.chula.ac.th

Keywords: Web page categorization, Iterative Cross-Training, Feature sets

Abstract: The paper presents the effects of different feature sets on the Web page categorization problem. These features are words appearing in the content of a Web page, words appearing on the hyperlinks, which link to the page and words appearing on every headings in the page. The experiments are conducted using a new algorithm called the Iterative Cross-Training algorithm (ICT) which was successfully applied to Thai Web page identification. The main concept of ICT is to iteratively train two sub-classifiers by using unlabeled examples in crossing manner. We compare ICT against supervised naïve Bayes classifier and Co-Training classifier. The experimental results show that ICT obtains the highest performance and the heading feature is considerably succeed in helping classifiers to build the correct model used in the Web page categorization task.

1. INTRODUCTION

Nowadays, there is a massive increase of Web pages in the Internet. An ideal search engine should have the most updated information of all Web pages to provide the best search result for the user. Therefore, it should have an effective Web robot which crawls the Web and automatically classifies Web pages into categories, since Web page classification task is a tedious job and time consuming process if it is done by human. Thus, we want it to be automatic with a reliable classification result.

The problem of text classification has been explored by many researchers with variety of learning algorithms (Cohen & Singer, 1999; Jochim, 1998) When we give a sufficient set of labeled training examples, supervised learning is the most effective method for the classification. However, the construction of hand-labeled data must be done by a human and thus this is a painfully time-consuming process.

Though it is costly to construct hand-labeled data, in some domains it is easy to obtain unlabeled ones, such as data in the Internet. Therefore, we propose a new learning algorithm called incremental iterative cross-training (incremental-ICT) in order to utilize the available unlabeled data.

Our incremental-ICT is based on the ICT algorithm that has been successfully applied for identifying Thai Web pages (Kijsirikul et al., 2000). ICT employs two sub-classifiers to iteratively train each other by using unlabeled examples in crossing manner. ICT is based on the assumption that one of the sub-classifiers has some knowledge about the domain. However, this assumption is violated on some domains where we cannot give domain knowledge to the classifier. In such a problem, ICT does not perform well. In this paper, we propose a new algorithm, called incremental-ICT, which requires no such assumption.

To evaluate the robustness of our algorithm, we apply it to a more difficult problem than Thai Web page identification. The problem we are interested in is the classification of Web pages into course or non-course pages. Since the concept of ICT needs two classifiers, we build each classifier based on different feature sets using naïve Bayes classifiers.

We run experiments to evaluate the effectiveness of our method and to see the contribution of each feature set. In the experiments, we compare our method with the *Co-Training* algorithm (Blum & Mitchell, 1998) and a supervised learning algorithm which uses a naïve Bayes classifier as a classification mechanism. The results show that incremental-ICT gives better performance than the other classifiers.

The paper is organized as follows. Section 2 presents feature sets used in the experiments. Section 3 describes our learning algorithm, and gives the details of a naïve Bayes classifier. Section 4 and 5 describes other learning methods used in our comparison. Section 6 describes the experimental results. Finally, Section 7 concludes our work.

J. Filipe et al. (eds.), Enterprise Information Systems III, 132–138.

2. FEATURE SETS

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

2.1 Hyperlink

The Web page in the Internet is a special document. It has a unique characteristic which makes it different from other plain text documents. Most Web pages have hyperlinks that act like a pointer pointing to other pages and also have links from other pages pointing to them. In our case, we use the hyperlink which link to the page to be the first feature set.

2.2 Content

The content of a Web page provides information to the user in detail. We extract all words in the content to be the second feature set.

2.3 Heading

The heading phrase normally represents the main idea of the following content. We use this opportunity to extract all headings in the page in the hope that they could represent the main concept of a Web page.

3. INCREMENTAL ITERATIVE CROSS-TRAINING

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

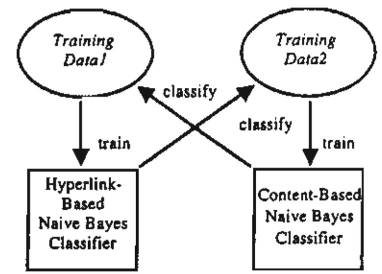


Figure 1: The architecture of iterative cross-training

The training algorithm of incremental-ICT is shown in Table 1. As shown in the table, the training process starts with the parameter estimation of both classifiers, i.e. hyperlink-based and content-based, using initial labeled data. For each round of iteration, the content-based classifier with the current parameter, θ_c, will classify training data into positive and negative examples. Then it will ask for the confirmation from the hyperlink-based classifier that considers another view of each example to make decision about which class the example should be. If both classifiers agree with the same classification

result, the most confident p positive and n negative examples will be labeled.

Table 1: The Incremental-ICT algorithm

Given:

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

The hyperlink-based classifier is then trained by the labeled examples in *TrainingData1* to estimate the parameter set θ_h. With this current θ_h, the hyperlink-based classifier will classify *TrainingData2* into positive and negative examples. Then the consistency checking process is performed again to ask for the agreement from the content-based classifier. The most confident p positive and n negative examples will be labeled. The content-based classifier starts again with parameter estimation by using labeled examples in *TrainingData1*. These processes will be repeatedly done until all data are labeled.

The classification mechanisms of these two classifiers ar the same which use the naïve Bayes algorithm. This algorithm is a well-known approach and is considered to be one of the most effective way for text classification (Mitchell, 1997). The algorithm employs *bag-of-words* to represent the document. The method is described below.

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

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

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

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

For our data set, L is the set of positive and negative class labels which are course homepage and non-course homepage, respectively. $Pr(w_1,\ldots, w_n)$ in equation 2 can be ignored, as we are interested in finding the most likely class label. As there are usually an extremely large number of possible values for $d = (w_1, w_2, \ldots, w_n)$, calculating the term $Pr(w_1,\ldots,w_n|l_j)$ requires a huge number of examples to obtain reliable estimation. Therefore, to reduce the number of required examples and improve reliability of the estimation, assumptions of naïve Bayes are made. These assumptions are (1) the conditional independent assumption, i.e. the presence of each word is conditionally independent of all other words in the document given the class label, and (2) an assumption that the position of a word is unimportant, e.g. encountering the word "subject" at the beginning of a document is the same as encountering it at the end (Mitchell, 1997). Equation 3 can be rewritten as:

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

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

The probabilities $Pr(l_j)$ and $Pr(w_i|l_j)$ are used as the parameter sets θ_h and θ_c, and are estimated from the training data. The prior probability $Pr(l_j)$ is estimated as the ratio between the number of examples belonging to the class l_j and the number of all examples. The conditional probability $Pr(w_i|l_j)$, of seeing word w_i given class label l_j, is estimated by the following equation:

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

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

To evaluate our method, we will compare it with the other two techniques that are the Co-Training and the supervised naïve Bayes classifiers. These classifiers are described in the following sections.

4. CO-TRAINING CLASSIFIER

The Co-Training algorithm explicitly uses the split of the features when learning from labeled and unlabeled data. Its approach is to build the naïve Bayes classifier for each of the distinct feature sets. Each classifier is initialized using a few labeled documents. Then every round of Co-Training, each classifier chooses the most confident p positive and n negative labeled examples to add to the labeled set of documents. The documents selected are those that have the highest posterior class probability, $Pr(l_j|d)$. Then, each classifier rebuilds from the augmented labeled set and the process repeats (Blum & Mitchell, 1998).

Table 2: The Co-Training algorithm

Given:

A set *LE* of labeled training examples

A set *UE* of unlabeled examples

Create a pool *UE'* of examples by choosing u examples at random from *UE*.

Loop while there exist documents without class labels:

- Use *LE* to estimate θ_h of the hyperlink-based classifier using the hyperlink portion of each document.
- Use *LE* to estimate θ_c of the content-based classifier using the page portion of each document.
- Allow the hyperlink-based classifier with current θ_h to label p positive and n negative examples from *UE'*.
- Allow the content-based classifier with current θ_c to label p positive and n negative examples from *UE'*.
- Add these self-labeled examples to *LE*.
- Randomly choose $2p+2n$ examples from *UE* to replenish *UE'*.

5. SUPERVISED NAÏVE BAYES CLASSIFIER

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

6. EXPERIMENTAL RESULTS

In order to test the robustness of the incremental-ICT algorithm and to investigate the effectiveness of feature sets , we set up experiments on the problem of course/non-course Web page

classification, and compare the performance of incremental-ICT to the other classifiers, i e , the Co-Training algorithm and the supervised naïve Bayes classifier.

6.1 Data Set

The data for our experiments is obtained via ftp from Carnegie Mellon University (The World Wide Knowledge Base Project). It consists of 1,051 Web pages collected from Computer Science department Web sites at four universities: Cornell, University of Washington, University of Wisconsin, and University of Texas. These Web pages have been hand-labeled into two categories. We consider the category "course home page" as the positive class and the other as the negative class. In this dataset, 22% of the Web pages were course homepages and the rest were non-course homepages.

In this data set, the two Web categories are actually closely related which make the classification more difficult. A course home page gives information about the subject such as the course outline, the class schedule, reference books. A non-course homepage is an instructor homepage or department Web page.

6.2 Experimental Setting

We have 230 course Web pages and 821 non-course Web pages. Each sample is filtered to remove words which give no significance in predicting to the class of the document. Words to be eliminated are auxiliary verbs, prepositions, pronouns, possessive pronouns, phone numbers, digit sequences, dates and special characters. The training set contains 172 course Web pages and 616 non-course Web pages.

Three positive examples and nine negative examples were randomly selected from the training dataset to be initial labeled data. Therefore, each data set contains 12 initial labeled examples, 776 training examples and 263 testing examples. We then used 3-fold cross-validation (Mitchell, 1997) for averaging the results. Three positive and nine negative samples are used as the initial labeled data for the incremental-ICT and the Co-Training algorithms. The parameters p and n in Table 1 and Table 2 is set to 1 and 3, respectively.

6.3 The Results

Standard precision (P), recall (R), accuracy (A) and F1-measure (F1) are used to evaluate the performance of the classifiers. These are defined as follows.

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

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

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

$$F1 = \frac{2PR}{P+R}$$

6.4 Experiment using content and hyperlink features

For the first experiment, we use words appearing in the content of a Web page as the feature for the first classifier. The second classifier uses words appearing on the hyperlink as a feature set. The results are shown in Table 3.

Table 3: Performance of content-based and hyperlink-based classifiers using 3-fold cross-validation: P = Precision, R = Recall, A = Accuracy, F1 = F1-measure.

Classifier	P(%)	R(%)	A(%)	F1
I-ICT (content)	94.04	80.46	94.55	86.72
S-Bayes (hyperlink)	85.34	63.22	89.48	72.61
I-ICT (hyperlink)	67.54	72.41	85.17	69.89
Co-Training (content)	81.52	54.08	87.32	65.03
S-Bayes (content)	99.05	42.20	87.25	58.97
Co-Training (hyperlink)	75.92	44.83	84.28	56.37

In Table 3, I-ICT (content) and I-ICT (hyperlink) stand for the content-based and hyperlink-based naïve Bayes classifiers of the incremental-ICT, respectively. Co-Training (content) and Co-Training (hyperlink) are content-based and hyperlink-based naïve Bayes classifiers of the Co-Training algorithm, respectively. S-Bayes (content) and S-Bayes (hyperlink) are supervised naïve-Bayes classifiers, which classify Web pages based on words in Web pages and words in hyperlinks, respectively.

As shown in the table, I-ICT (content) gives the best performance followed by S-Bayes (hyperlink), I-ICT (hyperlink), Co-Training (content), S-Bayes(content) and Co-Training (content), respectively. The reason that I-ICT (content) gives better performance compared to S-Bayes is because I-ICT (content) cooperates with I-ICT (hyperlink) while S-Bays uses single classifier. The performance of I-ICT (hyperlink) is not as good as that of I-ICT (content). This is because hyperlinks contain fewer words and thus are less capable of building the accurate classifier. The training technique of I-ICT is also an effective way, as its performance is better than that of Co-Training, which uses a different training technique.

6.5 Experiment using content and heading features

In order to see the impact of the heading feature on the categorization problem, we did experiments using heading-based classifier with various learning algorithms.

As shown in Table 4, I-ICT (content) and I-ICT(heading) stand for the content-based and heading-based naïve Bayes classifiers of the incremental ICT algorithm. S-Bayes (heading) and S-Bayes (content) are supervised naïve Bayes classifiers based on heading and content features, respectively.

Co-Training(heading) and Co-Training (content) are the heading-based and content-based naïve Bayes classifiers of the Co-Training algorithm.

Table 4: Performance of heading-based and content-based classifiers using 3-fold cross-validation: P = Precision, R = Recall, A = Accuracy, F1 = F1-measure.

Classifier	P(%)	R(%)	A(%)	F1
I-ICT (content)	98.12	78.66	95.05	87.32
S-Bayes (heading)	96.51	77.58	94.43	86.02
I-ICT (heading)	80.72	89.65	92.65	84.95
Co-Training (heading)	79.71	74.71	90.24	77.13
Co-Training (content)	82.49	43.68	85.93	57.11
S-Bayes(content)	99.05	42.20	87.25	58.97

The best performance belongs to the content-based naive Bayes classifier of I-ICT followed by the supervised naïve Bayes classifier based on the heading feature, the heading-based of I-ICT, the heading-based of co-training, the content-based of co-training and the content-based of the supervised naïve Bayes classifier.

7. DISCUSSION AND CONCLUSION

In this paper, we have demonstrated the concept of the I-ICT algorithm and investigate the impacts of feature sets to the classification correctness. From the experimental results, the performance of S-Bayes (heading) is much higher than that of S-Bayes (hyperlink). It means that the heading feature has more potential than the hyperlink feature in helping the classifier to build the correct model used in categorization task. This is because the detail of the Web page is usually organized into sub-sections with the headings, which represent the main idea of the following content. Thus the structure of all headings in a page should give some common properties that is useful to identify its category. In the contrary, the words in the hyperlink, which links to the page could not provide enough information to identify the class of the page. This is not surprising because normally the hyperlink phase contains just few words referring to the page. For the worst case, the hyperlink might contain only a proper noun that is not sufficient in classifying that referring page.

According to the F1-measure, we obtained only 58.97% accuracy for S-Bayes using the content feature. It implies that the content feature alone could not help much in Web page classification because the two Web categories actually have high relevance in detail. Therefore the naïve Bayes classifier could not find the exact model for each category using all words appearing in the page.

Considering all classification mechanisms, we found that our I-ICT algorithm provides the highest correctness in both experiments. This is because I-ICT combines two classifiers based on different feature sets and these two classifiers cooperate with each other during the training process. The I-ICT algorithm has been proved to be robust under new assumption that each example can be viewed in two different views using different feature sets. With the consistency checking process, which is used to compensate the lack of domain's knowledge of the classifiers, our algorithm outperformed the supervised naïve Bayes algorithm, and Co-Training algorithm.

ACKNOWLEDGEMENT

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

REFERENCES

Kijsirikul, B., Sasipongpairoege, P., Soonthornphisaj, N. and Meknavin, S., 2000, 'Supervised and Unsupervised Learning Algorithms for Thai Web Pages Identification', *Proceeding of the Pacific Rim*

International Conference on Artificial Intelligence (PRICAI-2000), 690-700

Blum, A. and Mitchell, T., 1998, 'Combining Labeled and Unlabeled Data with Co-Training', *Proceeding of the Eleventh Annual Conference on Computational Learning Theory*.

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

Joachims, T., 1998, 'Text categorization with support vector machines: Learning with many relevant feature', *Proceedings Tenth European Conference on Machine Learning*, Springer Verlag

Nigam, K., McCallum, A., Thrun, S and Mitchell, T., 2000, 'Text classification from labeled and unlabeled documents using EM', *Machine Learning*, 39(2/3): 103-134.

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

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

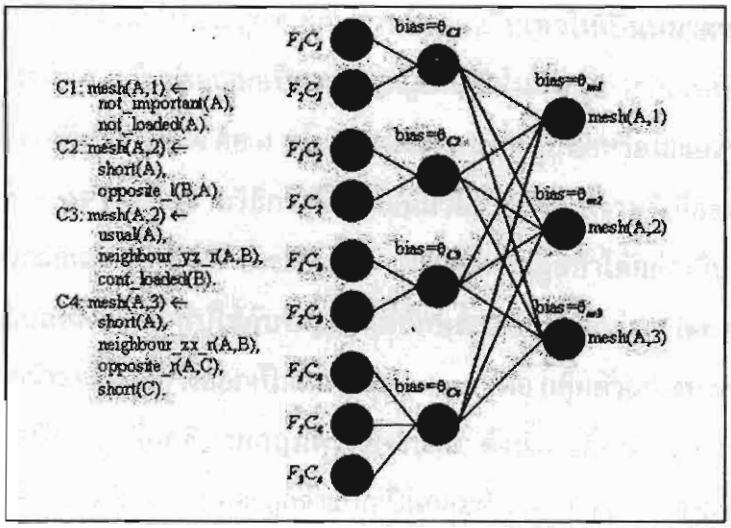
The World Wide Knowledge Base (web-kb) project, http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-51/www/co-training/data/course-cotrain-data.tar.gz, Carnegie Mellon University, U.S.A.

เอกสารหมายเลข 5

รางวัลผลงานวิจัย
จุฬาลงกรณ์มหาวิทยาลัย
ประจำปี ๒๕๔๕

รางวัลผลงานวิจัย
กองทุนรัชดาภิเษกสมโภช จุฬาลงกรณ์มหาวิ
ประจำปี ๒๕๔๕

ISBN 974-13-2291-7



ผลงานวิจัยเรื่อง การทำเหมืองเว็บไทยโดยเทคนิคการเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัย

Thai Web Mining Using Machine Learning and Inductive Logic Programming

โดย ผู้ช่วยศาสตราจารย์ ดร.บุญเสริม กิจศิริกุล

แหล่งทุนที่ได้รับ ทุนพัฒนานักวิจัย จากสำนักงานกองทุนสนับสนุนการวิจัย

5.3

งานวิจัยเรื่อง การทำเหมืองเว็บไทยโดยเทคนิคการเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัยนี้ มุ่งเน้นที่จะสร้างระบบค้นหาที่สามารถจัดหมวดหมู่ของเว็บเพจภาษาไทยให้ได้อย่างอัตโนมัติ ซึ่งเทคนิคที่จะนำมาใช้คือ การเรียนรู้ของเครื่องและการโปรแกรมตรรกะเชิงอุปนัย ผู้วิจัยได้ทำการวิจัยโดยเริ่มจากการดำเนินการวิจัยพื้นฐานเกี่ยวกับการโปรแกรมตรรกะเชิงอุปนัย ซึ่งจะนำไปประยุกต์ใช้กับการจำแนกเว็บเพจให้เป็นหมวดหมู่ต่อไป

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

ดังนั้นในงานวิจัยนี้จึงเป็นการมุ่งศึกษาหาวิธีการ ซึ่งสามารถนำมาใช้จำแนกตัวอย่างเพื่อใช้ร่วมกับกฎที่ได้จากระบบไอแอลพี เราได้ใช้กระบวนการดึงลักษณะสำคัญและวิธีการ

5.4

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

ผลการวิจัยที่ได้แสดงให้เห็นว่า การนำวิธีการดึงลักษณะสำคัญมาใช้ร่วมกับแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก ทำให้ประสิทธิภาพของการโปรแกรมตรรกะเชิงอุปนัยดีขึ้น สามารถจำแนกตัวอย่างที่ไม่ตรงพอดีกับกฎดั้งเดิมได้เป็นอย่างดี และผลการทดลองที่ได้แสดงให้เห็นถึงเปอร์เซ็นต์ความถูกต้องที่เพิ่มขึ้นของวิธีการที่นำเสนอ เมื่อเทียบกับระบบไอแอลพีอื่นที่มีอยู่ โดยได้ทำการทดลองกับชุดข้อมูลหลายชุด และวิธีการที่นำเสนอให้ผลที่ดีกว่าระบบไอแอลพีที่มีอยู่ทุกระบบในทุกชุดข้อมูล

5.5

สิ่งที่ดีเด่นในผลงานวิจัย

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

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

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

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

5.6

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

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

Nuanwan Soonthornphisaj[1] and Boonserm Kijsirikul[2]

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

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

1. Introduction

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

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

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

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

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

2. Incremental Iterative Cross-Training

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

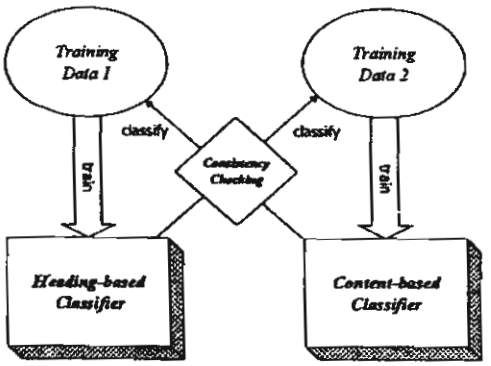


Figure 1 : The architecture of Incremental Iterative Cross-Training.

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

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

Table 1. Incremental-ICT algorithm

Given:

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

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

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

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

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

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

For our data set, L is the set of class labels, which are the categories of Web pages that we want the classifier to learn their concepts. $Pr(w_1, \ldots, w_n)$ in equation 2 can be ignored, as we are interested in finding the most likely class label. As there are usually an extremely large number of possible values for $d = (w_1, w_2 \ldots, w_n)$, calculating the term $Pr(w_1, \ldots, w_n|l_j)$ requires a huge number of examples to obtain reliable estimation. Therefore, to reduce the number of required examples and improve reliability of the estimation, assumptions of naive Bayes are made. These assumptions are (1) the conditional independent assumption, i.e. the presence of each word is conditionally independent of all other words in the document given the class label, and (2) an assumption that the position of a word is unimportant, e.g. encountering the word "subject" at the beginning of a document is the same as encountering it at the end [10]. Equation 3 can be rewritten as:

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

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

The probabilities $Pr(l_j)$ and $Pr(w_i|l_j)$ are used as the parameter sets θ_h and θ_c of the classifiers, and are estimated from the training data. The prior probability $Pr(l_j)$ is estimated as the ratio between the number of examples belonging to the class l_j, and the number of all examples. The conditional probability $Pr(w_i|l_j)$, of seeing word w_i given class label l_j, is estimated by the following equation:

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

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

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

3. Supervised Naive Bayes Classifier

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

4. Experimental Results

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

4.1. The result on WebKb data set

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

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

Experimental Setting

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

The settings for the classifiers are as follow.

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

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

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

The Results

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

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

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

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

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

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

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

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

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

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

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

6.4

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

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

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

4.2 The result on WebClass data set

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

Experimental Setting

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

The settings for classifiers are as follows.

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

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

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

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

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

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

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

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

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

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

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

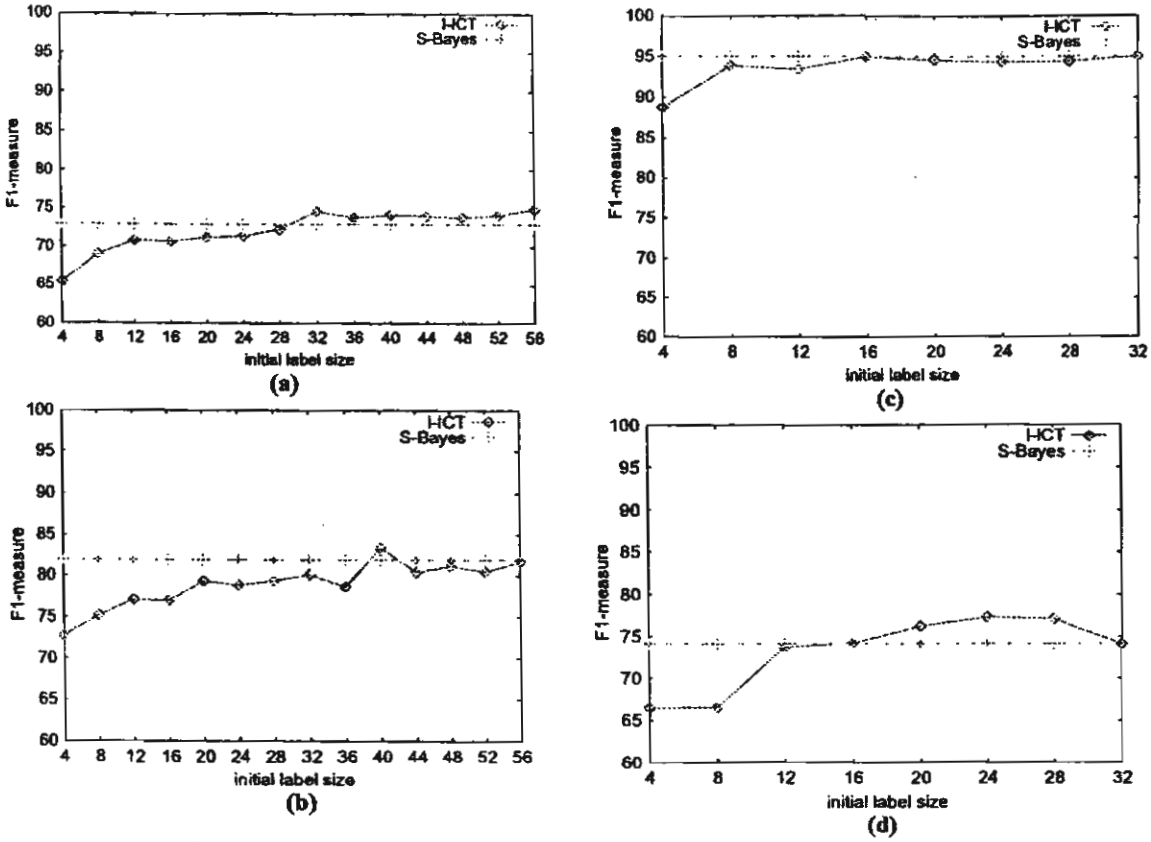


Figure 2. Performance of I-ICT on WebKb data set using content feature (a)
Performance of I-ICT on WebKb data set using heading feature (b)
Performance of I-ICT on WebClass data set using content feature (c)
Performance of I-ICT on WebClass data set using heading feature (d)

5. Discussion

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

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

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

6. Conclusion

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

Acknowledgement

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1. Introduction

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

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

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

7.1

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

2. Approximate ILP Rules by a Backpropagation Neural Network

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

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

2.1. Feature Generation

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

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

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

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

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

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

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

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

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

The examples of closed and open chains are shown below.

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

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

some closed chains are:

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

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

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

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

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

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

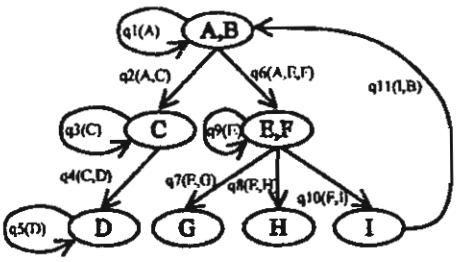


Figure 1: The dependency graph for the rule "p(A,B)←q1(A), q2(A,C), q3(C), q4(C,D), q5(D), q6(A,E,F), q7(E,G), q8(E,H), q9(E), q10(F,I), q11(I,B)."

Table 1: The algorithm for feature generation.

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

7.3

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

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

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

2.2. Building Network Structure from Features

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

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

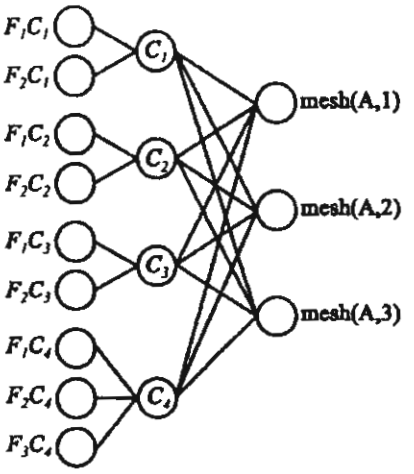


Figure 2: The structure of the neural network for the rule set $\{C_1, C_2, C_3, C_4\}$ in Section 2.2.

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

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

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

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

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

2.3. Training the Network

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

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

3. Experiments & Results

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

3.1. DataSets

Thai Character Recognition (TCR)

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

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

Finite Element Mesh Design (FEM)

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

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

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

7.5

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

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

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

Mutagenesis

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

King-Rook-King Chess Endgame (KRK)

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

3.2. Experimental Results & Discussions

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

The results show that the performance of PRO-

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

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

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

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

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

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

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

3.3. An Additional Experiment on KRK Noisy Datasets

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

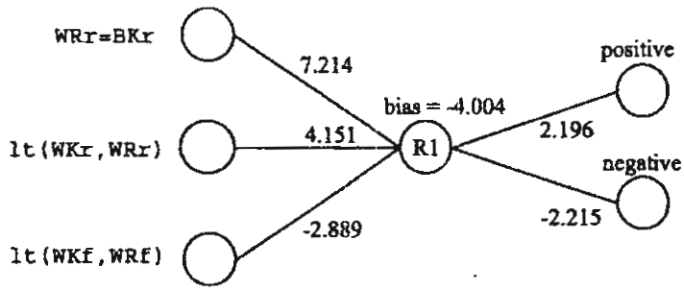


Figure 3: The portion of network for the rule "illegal(WKf,WKr,WRf,WRr,BKf,BKr) ← WRr=BKr, lt(WKr,WRr), lt(WKf,WRf)."

negative, and vice versa).

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

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

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

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

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

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

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

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

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

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

4. Conclusions

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

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

Acknowledgement

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1. Introduction

Given pre-labeled training data, supervised learning has been successfully applied to text classification [1,3,4,6,7,9,16]. However, one of the difficulties of using supervised learning is that we have to hand-label data for constructing training sets. Though it is costly to construct hand-labeled data, in some domains it is easy to obtain unlabeled ones, such as data in the World Wide Web. Thus, if we are able to effectively utilize the available unlabeled data, we will simplify the task of building text classifiers. Various methods have been proposed to use unlabeled data together with pre-labeled data for text classification, such as *active learning with committee* [10], text classification using EM [14], *co-training* algorithm [2].

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

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

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

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

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

2. Iterative Cross-Training

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

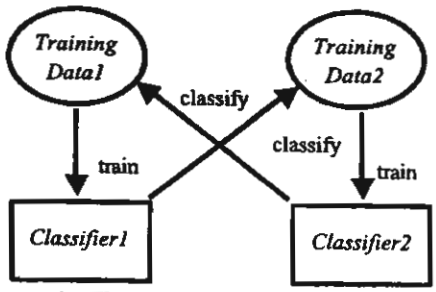


Figure 1: The architecture of Iterative Cross-Training. It is composed of two classifiers which use unlabeled data to iteratively train each other.

Figure 1 shows our learning system which learns to classify Web pages. The system is composed of two classifiers: *Classifier1* and *Classifier2*. Given domain knowledge or a small set of pre-labeled data, these two classifiers estimate their parameters from unlabeled data by receiving training from each other. Two training data sets, called *TrainingData1* and *TrainingData2* are duplicated from the unlabeled data provided by the user. Let θ_1 and θ_2 be sets of parameters of *Classifier1* and *Classifier2*, respectively. *TrainingData1* is used to train *Classifier1* to estimate its parameter set, and the *TrainingData2* is used to estimate the parameter set of *Classifier2*. The algorithm for training the classifiers is shown in Table 1.

The idea behind our algorithm is that if we can obtain reliable statistical information contained in *TrainingData2*, it should be useful in classifying *TrainingData1*. If the starting parameter set of *Classifier1* (θ_{1_0}) has property that it produces more true positive than wrong positive and more true negative than wrong negative examples for *TrainingData2*, the statistical information in correctly classified examples will be obtained.

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

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

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

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

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

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

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

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

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

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

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

The following subsections describe the details of the classifiers.

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

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

(1) Word Segmentation Classifier (*Classifier1*)

One straightforward way to determine whether a Web page is in a specific language is to check the words in the page with a dictionary. If *many* words appear in the dictionary, it is likely that the page is in that language. We cannot hope that all words in the page appear in dictionary as the Web page usually contains names of persons, organizations, etc. not occurring in the dictionary and may contains words written in foreign languages. Therefore, it is necessary to determine how many words should be contained. This task is more difficult when it is considered in a language that has no word boundary delimiters, such as Thai, Japanese, etc. [12].

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

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

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

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

In the following experiments, $\eta 1$ and $\eta 2$ are set to 1 and 2, respectively. As generating all possible segmentations and calculating their costs is very expensive, we employ dynamic programming technique to implement this calculation. Note that any sequence of characters, $c_i,\ldots,c_j$, found in the dictionary must be considered as a word, and must not be grouped with nearby characters to form a long unknown string.

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

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

Given sets of positive and negative examples, the classifier finds the threshold of *WordRatio* that maximizes the number of correctly classified positive and negative examples. If *WordRatio* of a page is greater than the threshold, we will classify it as positive (Thai page). Otherwise, we will classify it as negative (non-Thai page). For simplicity, let us use only the threshold of *WordRatio* as the parameter of word segmentation classifier (θ_i).

Having only the threshold of *WordRatio* (θ_i) as the parameter, we can find θ_{i0} which produces more true positive and true negative examples for *TrainingData2*. As describes above, most of Thai pages should have a high value of *WordRatio*, whereas non-Thai pages should have a low value one. If the numbers of Thai and non-Thai pages in *TrainingData2* are the same, it is easily to see that any value of θ_{i0} will give more correctly classified pages than incorrectly ones (except for $\theta_{i0} = 0.0$ or $\theta_{i0} = 1.0$, that gives the same number of correctly and incorrectly classified pages). In case that the number of Thai pages is lower than the number of

non-Thai pages, a high value of θ_0, (e.g. 0.7, 0.8, 0.9) will produce more correctly classified pages. This is the case that is likely to be encountered in the real world. A low value of θ_0 is for the case that the number of Thai pages is larger than that of non-Thai pages.

A new θ can be estimated, after the naï ve Bayes classifier (*Classifier2*) labels data in *TrainingData1*. Let *SP* be the smallest value of *WordRatio*'s from all labeled positive examples, and *LN* be the largest value from all labeled negative examples. In case of $SP \geq LN$, the new θ is estimated as:

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

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

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

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

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

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

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

For text classification, naï ve Bayes is among the most commonly used and the most effective methods [13]. To represent text, the method usually employs *bag-of-words* representation. Instead of bag-of-words, we use the simpler *bag-of-characters* representation in the problem of classification of Thai/non-Thai pages. This representation is suitable for a Web robot to identify Thai Web pages, because it requires no word segmentation and thus it is very fast. In spite of its simplicity, our results show the effectiveness of bag-of-characters representation in identifying Thai Web pages, as shown later in Section 4.

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

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

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

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

In our case, L is the set of positive and negative class labels. The term $\Pr(c_1,\ldots,c_n)$ in Equation 4 can be ignored, as we are interested in finding the most likely class label.

As there are usually an extremely large number of possible values for $d = (c_1,c_2,\ldots,c_n)$, calculating the term $\Pr(c_1,c_2,\ldots,c_n \mid l_j)$ requires a huge number of examples to obtain reliable estimation. Therefore, to reduce the number of required examples and improve reliability of the estimation, assumptions of naï ve Bayes are made [13]. These assumptions are (1) the conditional independent assumption, i.e. the presence of each character is conditionally independent of all other characters in the document given the class label, and (2) an assumption that the position of a character is unimportant, e.g. encountering the character "a" at the beginning of a document is the same as encountering it at the end. Clearly, these assumptions are violated in real-world data, but empirically naï ve Bayes has successfully been applied in various text classification problems [7,11,17].

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

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

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

This model is also called *unigram* model because it is based on statistics about single character in isolation.

The probabilities $\Pr(l_j)$ and $\Pr(c_i|l_j)$ are used as the parameter set θ_2 of our naï ve Bayes, and are estimated from the training data. The prior probability $\Pr(l_j)$ is estimated as the ratio between the number of examples belonging to the class l_j and the number of all examples. The conditional probability $\Pr(c_i|l_j)$, of seeing character c_i given class label l_j, is estimated by the following equation:

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

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

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

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

3. Other Classifiers Used in Comparison

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

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

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

Co-Training-Style Classifier

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

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

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

Table 2: The co-training-style algorithm.

Given:
- a set *LE* of labeled training examples
- a set *UE* of unlabeled examples

Create a pool *UE'* of examples by choosing u examples at random from *UE*

Loop until no examples left in *UE*:
- Use *LE* to estimate the parameter set θ_1 of *Classifier1*.
- Use *LE* to estimate the parameter set θ_2 of *Classifier2*.
- Allow *Classifier1* with θ_1 to label p positive and n negative examples from *UE'*.
- Allow *Classifier2* with θ_2 to label p positive and n negative examples from *UE'*.
- Add these self-labeled examples to *LE*
- Randomly choose $2p+2n$ examples from *UE* to replenish *UE'*

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

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

4. Experimental Results

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

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

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

Data Set & Experimental Setting

We collected the data set by starting from four Web pages: a Japanese Web page[1], two Thai Web pages[2], and an English web page[3]. From each of these four pages, a Web robot was used to recursively follow the links within the page until it retrieves 450 pages. Therefore, we have approximately 900 Thai pages as Thai pages may link to ones which are in English or other languages. We also have approximately 450 Japanese and 450 English pages. All of these pages were divided into three sets, denoted as A, B and C, each of which contains 600 pages (about 300 Thai, 150 Japanese and 150 English pages). Note that HTML mark-up tags were removed before training and testing process. We used 3-fold cross validation in all experiments below for averaging the results.

The settings for the classifiers are as follows.

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

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

The Results

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Data Set & Experimental Setting

The data for our experiment is obtained via ftp from

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

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

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

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

The Results

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

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

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

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

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

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

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

5. Discussion and Related Work

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

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

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

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

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

6. Conclusion

We have presented a method that effectively uses unlabeled examples to estimate the parameters of the system for classifying Web pages. The method is based on two sub-classifiers that iteratively train each other. With no pre-labeled or a small set of pre-labeled examples, our method gives high precision and recall on classifying Web pages. The performance of our method is competitive with those of supervised ones, which demonstrates the successful use of unlabeled data of our method.

Acknowledgement

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8.9

International Conference on Information Technology Interface
ITI-2002, 2002

Web Page Categorization using Hierarchical Headings Structure

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

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

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

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

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

1. Introduction

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

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

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

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

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

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

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

2. Feature Sets

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

2.1 Content

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

2.3 Hierarchical Headings

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

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

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

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

3.Incremental Iterative Cross-Training

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

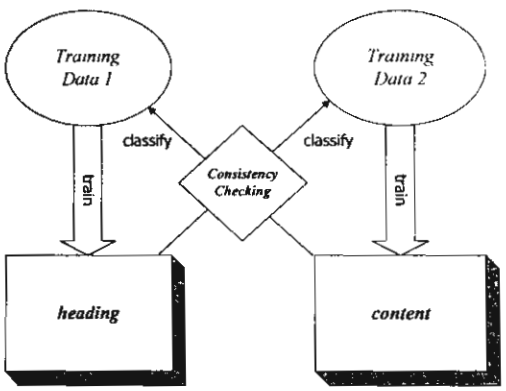


Figure 1. The architecture of Incremental Iterative-Cross Training algorithm.

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

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

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

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

Table 2. Incremental - ICT algorithm

Given:

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

For our data set, L is the set of class labels $Pr(w_1, \ldots, w_n)$ in equation 2 can be ignored, as we are interested in finding the most likely class label. As there are usually an extremely large number of possible values for $d = (w_1, w_2, \ldots, w_n)$, calculating the term $Pr(w_1, \ldots, w_n|l_j)$ requires a huge number of examples to obtain reliable estimation. Therefore, to reduce the number of required examples and improve reliability of the estimation, assumptions of naive Bayes are made. These assumptions are (1) the conditional independent assumption, i.e. the presence of each word is conditionally independent of all other words in the document given the class label, and (2) an assumption that the position of a word is unimportant, e.g. encountering the word "subject" at the beginning of a document is the

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

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

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

The probabilities $Pr(l_j)$ and $Pr(w_i|l_j)$ are used as the parameter sets θ_h and θ_c, and are estimated from the training data. The prior probability $Pr(l_j)$ is estimated as the ratio between the number of examples belonging to the class l_j, and the number of all examples. The conditional probability $Pr(w_i|l_j)$, of seeing word w_i given class label l_j, is estimated by the following equation:

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

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

4. The Co-Training Algorithm

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

Table 3: The Co-Training algorithm

Given:
A set *LE* of labeled training examples
A set *UE* of unlabeled examples
Create a pool *UE'* of examples by choosing u examples at random from *UE*.
Loop while there exist documents without class labels:

- Use *LE* to estimate θ_h of the hyperlink-based classifier using the hyperlink portion of each document.
- Use *LE* to estimate θ_c of the content-based classifier using the page portion of each document.
- Allow the hyperlink-based classifier with current θ_h to label p positive and n negative examples from *UE'*.
- Allow the content-based classifier with current θ_c to label p positive and n negative examples from *UE'*.
- Add these self-labeled examples to *LE*.
- Randomly choose $2p+2n$ examples from *UE* to replenish *UE'*.

5. Supervised Naive Bayes Algorithm

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

6. Experimental Results

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

different learning algorithms are shown in the next section.

6.1 University-related Data Sets

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

6.2 Drug-Usage data set

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

6.3 Experimental Setting

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

The Results

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

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

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

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

6.4 Experimental results of Supervised Naive Bayes Classifier.

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

Table 4. The F_1 performance on Drug Usage Data set

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

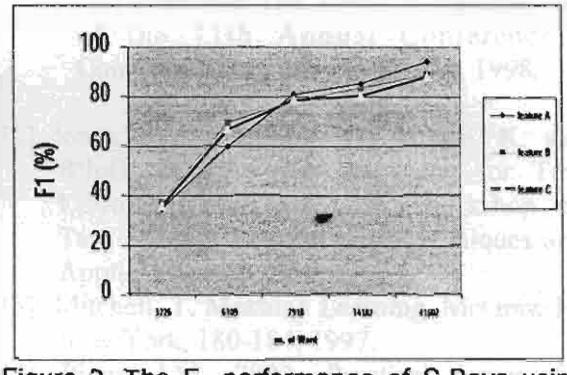
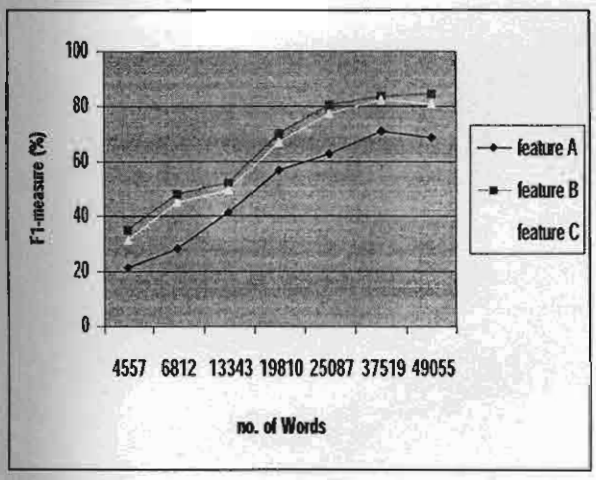


Figure 2. The F_1 performance of S-Bays using different no of words in the training data in the drug usage data set.

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

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

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



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

6.5 Experimental results using hierarchical headings structure on different algorithms.

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

Table 6. The F_1 performance on Drug Usage Data set

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

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

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

7. Discussion

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

8. Conclusion

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

9. References

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

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

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

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

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

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