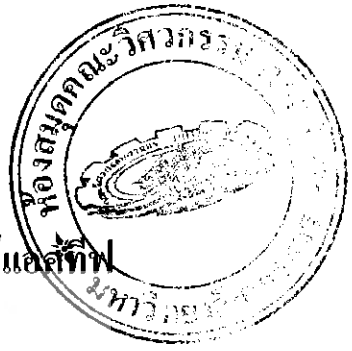


ระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ

Interactive Web Service Search Interface

นางสาวสุนันท์ ธาติ รหัส 50360364

นางสาวนภารัตน์ ปุ้มตะมะ รหัส 50361378

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2553

ใบรับรองปริญญานิพนธ์

หัวข้อโครงงาน	ระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ	
ผู้ดำเนินโครงงาน	นางสาวสุนันท์ ชาติ	รหัสนิสิต 50360364
	นางสาวนภารัตน์ ปุ้มตะมะ	รหัสนิสิต 50361378
อาจารย์ที่ปรึกษา	ดร.วรลักษณ์ คงเด่นฟ้า	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์	
ปีการศึกษา	2553	

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

.....ที่ปรึกษาโครงงาน
(ดร.วรลักษณ์ คงเด่นฟ้า)

.....กรรมการ
(นายภาณุพงศ์ สอนคม)

.....กรรมการ
(นายเศรษฐา ตั้งค้าวานิช)

หัวข้อโครงงาน	ระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ	
ผู้ดำเนินโครงงาน	นางสาวสุนันท์ ธาติ	รหัสนิสิต 50360364
	นางสาวนภารัตน์ ปุ้มตะมะ	รหัสนิสิต 50361378
อาจารย์ที่ปรึกษา	ดร.วรลักษณ์ คงเด่นฟ้า	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์	
ปีการศึกษา	2553	

.....

บทคัดย่อ

ในปัจจุบันนี้ การใช้งานเว็บเซอร์วิสกำลังแพร่หลายและได้รับความนิยมเพิ่มมากขึ้น แต่ในขณะเดียวกันระบบค้นหาเว็บเซอร์วิสยังมีจำนวนไม่มากนัก คณะผู้จัดทำจึงคิดริเริ่มในการสร้างระบบค้นหาเว็บเซอร์วิสขึ้น จากการศึกษาข้อมูลเกี่ยวกับการค้นหาข้อมูลของระบบ search engine พบว่า การค้นหาข้อมูลโดยใช้คำค้นหา(Keyword Search) มีปริมาณผู้ใช้งานมากที่สุด แต่การค้นหาข้อมูลโดยใช้คำค้นหา(Keyword Search) นั้น ผู้ใช้ไม่สามารถเลือกคำหรือประโยคค้นหาที่เหมาะสมให้แก่ระบบได้ ส่งผลให้ผลการค้นหาไม่ตรงตามความต้องการของผู้ใช้

ในปริญญานิพนธ์นี้ ได้กล่าวถึงการพัฒนาส่วนต่อประสานของระบบค้นหาเว็บเซอร์วิส (Web Service Search Interface) โดยเมื่อผู้ใช้กรอกคำหรือประโยคที่ต้องการค้นหา ระบบจะแสดงคำที่มีความหมายเหมือนกัน(Synonym) ให้ผู้ใช้เลือก พร้อมทั้งแสดงประโยคแนะนำซึ่งได้จากความสัมพันธ์ในออนโทโลยี (Ontology) ของคำค้นหา หลังจากนั้นเมื่อผู้ใช้เลือกคำค้นหาหรือประโยคที่ระบบแนะนำ ระบบจะสร้างนิพจน์ทางตรรกศาสตร์ (Boolean Expression) ซึ่งนิพจน์ทางตรรกศาสตร์ (Boolean Expression) ดังกล่าวเป็นเงื่อนไขในการค้นหาเว็บเซอร์วิส ระบบค้นหาเว็บเซอร์วิสที่พัฒนาขึ้นจะทำงานโต้ตอบกับผู้ใช้ตลอดเวลา เพื่อให้คำแนะนำในการค้นหาและแสดงผลการค้นหาในเวลาเดียวกัน ทำให้ผู้ใช้สามารถเปลี่ยนแปลงข้อมูลที่ใช้ในการค้นหาจนกระทั่งได้เว็บเซอร์วิสที่ตรงตามความต้องการของผู้ใช้มากที่สุด

Project Title	Interactive Web Service Search Interface		
Name	Miss.Sunun	Tati	ID. 50360364
	Miss.Naparat	Poomtama	ID. 50361378
Project Advisor	Woralak Kongdenfha, Ph.D.		
Major	Computer Engineering		
Department	Electrical and Computer Engineering		
Academic year	2010		

ABSTRACT

Web services are reusable software components that can deliver data and application functionalities over the internet. These services can then be discovered and integrated to develop new value-added applications. Web services are therefore the driving force for the evolution of internet from a network of static web pages to a dynamic and interactive application development platform. While web services bring a huge benefit to data and application development, the task of service discovery is still not well supported. Although the technology like the Universal Description, Discovery and Integration (UDDI), was invented with intentions to automate service discovery and integrations, it is difficult for service providers to provide descriptive information about published web services hence service searches would be inaccurate and not useful for searches. Moreover, the enormous amount of web search engines available on the internet today are not efficient for web services searches since web services typically lack of detailed descriptions required by most search methods. In this thesis, we therefore propose a specialized search engine to assist service requestors in their tasks of finding web services.

Based on our study about current web search engine technologies, we have found that the keyword and phrase-based searches are the most popular and success ways that users employed when finding information on the internet. Nevertheless, users often cannot think of appropriate keywords or phrases to use when searching for information they needed. We therefore developed an interactive service search interface that allows users to improve their service searches. With this interactive interface, users can immediately see the related set of web services and modify search keywords or phrases until an appropriate set of web services are obtained. Particularly, the interface suggests a set of alternatives that users may choose to improve their searches. These

suggestions are generated from WordNet and a set of domain-specific ontologies related to the search keywords. In cases of search phrases, we particularly adopt the Part-Of-Speech (POS) tagger to translate search phrases into a set of keywords before the suggestions are generated. We have conducted a user study on the developed service search interface and found that it eases users in the tasks of finding their required web services.

กิตติกรรมประกาศ

โครงงานเรื่อง ระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ (Interactive Web Service Search Interface) นี้ สำเร็จลุล่วงได้ด้วยดี ด้วยความกรุณาของ ดร.วรลักษณ์ คงเด่นฟ้า อาจารย์ที่ปรึกษาโครงงาน ซึ่งได้ทุ่มเททั้งกำลังกายและกำลังใจ ให้คำแนะนำ ข้อชี้แนะและความช่วยเหลือมาโดยตลอด ทั้งในส่วนของการออกแบบโปรแกรม วิธีการเขียนโปรแกรม การตรวจสอบโปรแกรม รวมทั้งช่วยตรวจสอบและให้คำแนะนำในการเขียนปริญญานิพนธ์ ขอขอบพระคุณ ดร.พงศ์พันธ์ กิจสนาโยธิน อ.ภาณุพงศ์ สอนคม และ อ.เศรษฐา ตั้งค้าวานิช กรรมการคุมสอบโครงงาน ที่ช่วยชี้แนะให้เห็นถึงข้อบกพร่องของโครงงาน ทำให้คณะผู้จัดทำได้แก้ไขโครงงานให้สมบูรณ์ยิ่งขึ้นไป

ขอขอบพระคุณครอบครัวที่เป็นแรงสนับสนุนอันสำคัญ รวมทั้งให้การดูแลและเอาใจใส่คณะผู้จัดทำเป็นอย่างดี

ทั้งนี้ คณะผู้จัดทำโครงงาน จึงขอบพระคุณทุกท่านที่มีส่วนช่วยเหลือให้งานนี้สำเร็จลุล่วงไปด้วยได้ดี ไว้ ณ โอกาสนี้

นางสาวสุนันท์ ธาติ

นางสาวนภารัตน์ ปุ้มตะมะ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

บทที่ 1

บทนำ

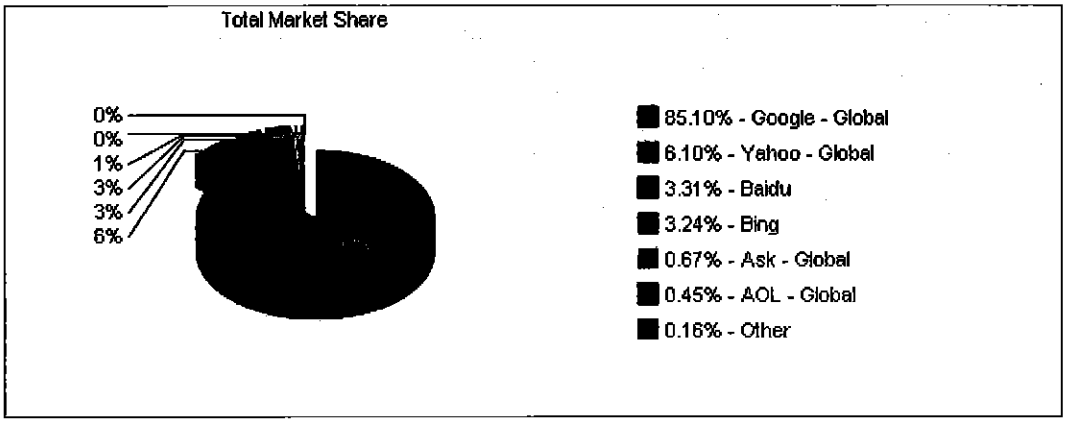
1.1 ที่มาและความสำคัญของโครงงาน

การค้นหาข้อมูลในยุคปัจจุบัน ถือเป็นสิ่งสำคัญอย่างยิ่ง เนื่องด้วยปัจจุบันมีการค้นหาข้อมูลทางอินเทอร์เน็ตมากขึ้น เนื่องจากในระบบอินเตอร์เน็ตมีข้อมูลที่หลากหลาย ทันสมัย และสะดวกและรวดเร็วต่อการเข้าถึงข้อมูล อีกทั้งในอนาคตการค้นหาข้อมูลทางอินเทอร์เน็ตยังมีแนวโน้มที่จะได้รับความนิยมมากขึ้นเรื่อยๆ จำนวนครั้งในการค้นหาข้อมูลมีปริมาณเพิ่มขึ้นอย่างรวดเร็ว ดังจะเห็นได้จากสถิติ

ตารางที่ 1.1 แสดงจำนวนครั้งในการค้นหาข้อมูล ของประเทศสหรัฐอเมริกาในเดือน มีนาคม

Searches	Per Day (Millions)	Per Month (Millions)
Google	91	2,733
Yahoo	60	1,792
MSN	28	845
AOL	16	486
Ask	13	378
Others	6	166
Total	**213**	**6,400**

จากสถิติปริมาณการค้นหาที่เพิ่มขึ้นอย่างรวดเร็ว เว็บไซต์สำหรับการค้นหาข้อมูลต่างๆจึงเกิดขึ้นเพื่อให้เพียงพอต่อความต้องการของผู้ใช้ เช่น Google, Yahoo และ Bing เป็นต้น



รูปที่ 1.1 แผนภูมิวงกลมแสดงส่วนทางการตลาดของ Search Engine ในปีค.ศ.2010

(Search Engine Market Share January, 2010 to December, 2010)

ในระบบการค้นหาข้อมูลมีหลากหลายรูปแบบ ซึ่งระบบค้นหาแบบอินเตอร์แอคทีฟ (Interactive Search) นั้น ถือเป็นอีกวิธีหนึ่งที่มีประสิทธิภาพและถูกนำมาใช้อย่างกว้างขวาง โดยระบบค้นหาแบบอินเตอร์แอคทีฟนี้ทำให้เกิดการปฏิสัมพันธ์กันระหว่างระบบการค้นหา (search engine) กับผู้ใช้ โดยระบบค้นหาแบบอินเตอร์แอคทีฟเป็นส่วนที่ติดต่อกับผู้ใช้นั่นเอง เป็นเสมือนส่วนประสานงานและรับข้อมูลที่ต้องการค้นหา (keyword search) จากผู้ใช้

ส่วนต่อประสานระบบค้นหาแบบอินเตอร์แอคทีฟ มีความสำคัญต่อการค้นหาข้อมูลในแต่ละครั้ง เนื่องจากความหมายของคำค้นหาของผู้ใช้ มีความกำกวม ดังนั้นผู้จัดทำจึงได้พัฒนาส่วนต่อประสานระบบค้นหาแบบอินเตอร์แอคทีฟที่สามารถแนะนำคำและประโยคแก่ผู้ใช้ เพื่อช่วยให้การค้นหาข้อมูลได้รวดเร็ว สะดวก และตรงกับความต้องการมากขึ้น

เนื่องด้วยปัจจุบันพบว่า เว็บเซอร์วิสได้รับความนิยมเป็นอย่างมาก เนื่องจากเว็บเซอร์วิสเป็นเทคโนโลยีที่ทำให้แอพพลิเคชั่นต่างๆสามารถสื่อสารแลกเปลี่ยนข้อมูลกันได้ ถึงแม้ว่าแอพพลิเคชั่นเหล่านั้นจะสร้างมาจากสถาปัตยกรรม ภาษา และฐานข้อมูลที่ต่างกันก็ตาม โดยมีการทำงานอยู่บนอินเตอร์เน็ตโปรโตคอล (Internet Protocol) ทั้ง HTML,TCP/IP โดยใช้ภาษา XML เป็นภาษาที่ทำการเข้ารหัสและถอดรหัสข้อมูลที่ส่งผ่านกันระหว่างไคลแอนต์กับเซิร์ฟเวอร์ เมื่อเว็บเซอร์วิสตัวใดตัวหนึ่งเริ่มทำงาน เว็บเซอร์วิสตัวอื่นก็สามารถรับรู้และเริ่มทำงานได้อีกด้วย โดยการที่สร้างฟังก์ชั่นตัวหนึ่งฝังไว้ในตัวเว็บแอพพลิเคชั่นและเว็บเซิร์ฟเวอร์ เพื่อที่จะให้ไคลแอนต์ หรือเว็บไซต์อื่น ๆ ที่สามารถเรียกใช้ฟังก์ชันนั้นๆได้ คณะผู้จัดทำเล็งเห็นประโยชน์และความสำคัญของเว็บเซอร์วิส จึงได้พัฒนาส่วนต่อประสานระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 สร้างและพัฒนาส่วนต่อประสานระบบค้นหาแบบอินเตอร์แอคทีฟ (Interactive search)

1.2.2 สามารถนำส่วนต่อประสานระบบค้นหาแบบอินเตอร์แอคทีฟ (interactive search) ไปใช้ร่วมกับ search enginc ได้

1.3 ขอบข่ายของโครงงาน

1.3.1 สร้างและพัฒนาส่วนต่อประสานระบบการค้นหาแบบอินเตอร์แอคทีฟ (Interactive search) โดยเป็นส่วนของเว็บแอบพลิเคชั่น เป็นอินเตอร์เฟส (Interface) ที่ติดต่อกันระหว่างระบบกับผู้ใช้

1.3.2 ส่วนต่อประสานระบบการค้นหาแบบอินเตอร์แอคทีฟสามารถปรับปรุงประโยคที่ผู้ใช้ต้องการค้นหา ให้สอดคล้องกับออนโทโลยี เพื่อสามารถตอบสนองการค้นหาข้อมูล (Search requirement) ได้ดีขึ้น

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาข้อมูลที่เกี่ยวข้องกับโครงสร้างออนโทโลยี

1.4.2 ศึกษาข้อมูลและทฤษฎีเกี่ยวกับ WordNet API

1.4.3 ศึกษาการทำงานของระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ

1.4.4 ศึกษาการทำงานของ FeedWinnover

1.4.5 ศึกษาการทำงานของโปรแกรม Stanford Log-linear Part-Of-Speech Tagger และหาแนวทางเพื่อเชื่อมต่อการทำงานกับระบบที่จะสร้างขึ้น

1.4.6 สร้างส่วนต่อประสานระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ

1.4.7 สร้าง tag cloud ของกลุ่มเว็บซอร์วิส

1.4.8 เขียนโปรแกรมเพื่อทดสอบระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ

1.4.9 ทดสอบการใช้งานโปรแกรมและแก้ไขข้อผิดพลาด

1.4.10 สรุปการดำเนินงานและเขียนปริญญานิพนธ์

1.5 แผนการดำเนินงาน

ตารางที่ 1.2 แผนการดำเนินงาน

รายละเอียด	ระยะเวลาการดำเนินงาน									
	ปี 2553							ปี 2554		
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1.ศึกษาข้อมูลที่เกี่ยวข้องกับการทำงานของออนโทโลยี										
2.ศึกษาข้อมูลและทฤษฎีเกี่ยวกับ wordNet API										
3.ศึกษาการทำงานของระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ										
4.ศึกษาการทำงานของ FeedWinnover และ โปรแกรม Stanford Log-linear Part-Of-Speech Tagger										
5.สร้างระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ										
6.สร้าง tag cloud ของ web serive										
7.เขียนโปรแกรมเพื่อทดสอบระบบการค้นหาเว็บเซอร์วิสแบบอิเตอร์แอคทีฟ										
8.ทดสอบการใช้งานโปรแกรมและแก้ไขข้อผิดพลาด										
9.สรุปการดำเนินงานและเขียนปริญญานิพนธ์										

1.6 ผลที่คาดว่าจะได้รับจากโครงงาน

1.6.1 ได้ระบบค้นหาแบบอินเตอร์แอคทีฟ (Interactive search) ที่สามารถใช้ร่วมกับ search engine ได้

1.6.2 ได้ระบบค้นหาแบบอินเตอร์แอคทีฟที่สามารถทำการค้นหาคำหลัก (Keyword) ได้อย่างครอบคลุม

1.7 งบประมาณ

1.7.1 ค่าหนังสือ	400 บาท
1.7.2 ค่าถ่ายเอกสาร	200 บาท
1.7.3 ค่าพัสดุสำนักงาน	100 บาท
1.7.4 ค่าจัดทำปริญญานิพนธ์	1300 บาท
รวมเป็นเงินทั้งสิ้น(สองพันบาทถ้วน)	2000 บาท

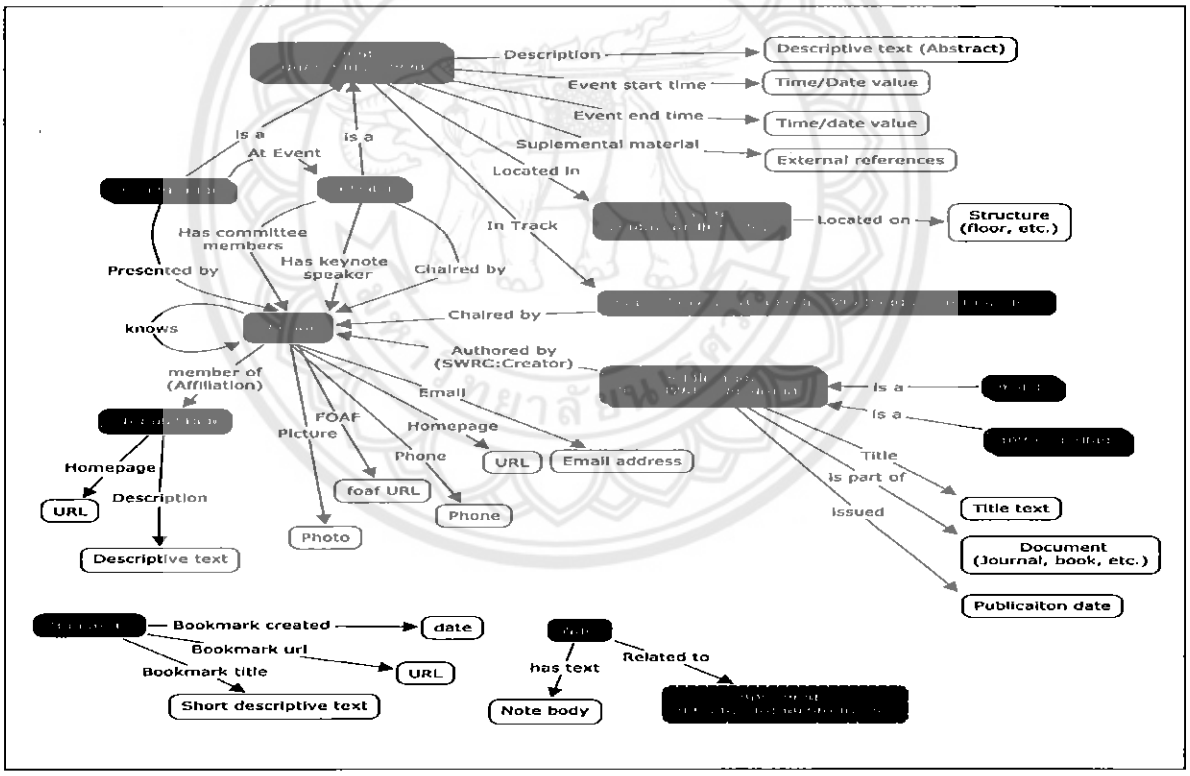
หมายเหตุ : ถัวเฉลี่ยทุกรายการ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ (Interactive Web Service Search Interface) เป็นระบบที่สร้างขึ้นเพื่อเป็นส่วนเว็บแอพพลิเคชั่นเป็นส่วนที่ติดต่อระหว่างระบบกับผู้ใช้ (Interface) และเพื่อการตอบสนองของ search requirement อย่างมีประสิทธิภาพสูงสุด

2.1 ศึกษาการทำงานของออนโทโลยี (Ontology)

จากการศึกษาตัวอย่างการทำงานของออนโทโลยี (Ontology) [1] เพื่อนำมาประยุกต์ใช้ในการสร้างระบบค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ



รูปที่ 2.1 แสดงตัวอย่างของ Ontology for semantic wiki

ออนโทโลยี (Ontology) ใช้ในการอธิบายโครงสร้าง และความสัมพันธ์ของระบบผ่านโหนดแบบลำดับชั้น ซึ่งมีรายละเอียดที่เกี่ยวข้องดังต่อไปนี้

2.1.1 Web Ontology Language (OWL)

Web Ontology Language (OWL) คือภาษาที่ใช้อธิบายออนโทโลยี ถูกสร้างโดย W3C (World Wide Web Consortium) Web Ontology Working Group โดย OWL ถูกพัฒนาขึ้นเพื่อเป็นส่วนขยายต่อจากภาษา Resource Description Framework (RDF) ภาษา OWL จัดเป็นองค์ประกอบหนึ่งในงานเว็บเชิงความหมาย (Semantic Web) ที่ใช้ในการบรรยายข้อมูล สามารถกำหนดโครงสร้างข้อมูลในลักษณะลำดับชั้น และอธิบายข้อมูล (Metadata) ที่มีความสัมพันธ์ในระบบฐานข้อมูลได้ รวมทั้งสามารถรองรับการบรรยายข้อมูลเชิงตรรกะ ชนิดข้อมูล และตัวบ่งปริมาณได้ ทำให้ข้อมูลที่ถูกแทนที่นั้นมีความหมายมากยิ่งขึ้น ลักษณะการบรรยายจะอยู่ในรูปของคลาส คุณสมบัติของคลาส และความสัมพันธ์ของคลาสเพื่ออธิบายเอนทิตี้และความสัมพันธ์ต่าง ๆ ที่เกิดขึ้น

Web Ontology Language หรือ "OWL" เป็นภาษาที่รวมกันระหว่างข้อความ (Text) และข้อความพิเศษ (Extra Information) ที่เพิ่มเติมเข้ามาเกี่ยวกับข้อความ ที่มีการอธิบายข้อมูลเป็นลำดับชั้นและความสัมพันธ์ระหว่าง Resources ที่แตกต่างกัน

2.2 ศึกษาการทำงานของ WordNet API

WordNet เป็นฐานข้อมูลที่รวบรวมคำศัพท์ภาษาอังกฤษไว้ ประกอบไปด้วยคำนาม (Nouns), คำกริยา (Verbs), คำคุณศัพท์ (Adjectives) และคำวิเศษณ์ (Adverbs) โดย WordNet สามารถค้นหาคำที่มีความหมายเหมือนกัน (synonym) ได้ โดยโครงสร้างของ WordNet ก็คือ

2.2.1 การค้นหาคำที่มีความหมายเหมือนกัน (Synonym)

Synonym คือ คำที่แตกต่างกันที่มีความหมายเหมือนกัน การค้นหาคำที่มีความหมายเหมือนกันทำได้โดย การหาความหมายนัยต่างๆของคำแต่ละคำ และนำความหมายเหล่านั้นมาหาคำที่มีความหมายเหมือนกัน

2.2.2 NHunspell

NHunspell เป็นชื่อ API ซึ่งทำงานได้หลายอย่าง ทั้งเช็คตัวสะกด หาคำที่มีความหมายเหมือนกัน มีความสามารถเป็น dictionary ได้ NHunspell ทำงานบนไมโครซอฟท์ .NET Framework เป็น C# library และ native library สำหรับ Hunspell, Hyphen และ MyThes เป้าหมายของการออกแบบของ library นี้คือจะเก็บข้อมูลในส่วนของ source code เวอร์ชันใหม่ของ base library จึงสามารถนำไปใช้กับ NHunspell ส่วน library จะใช้งานร่วมกับ OpenOffice และทำงานเกี่ยวกับ dictionary ที่ใช้งานกับ OpenOffice ซึ่ง NHunspell เป็นเจ้าของ license กับ GPL/LGPL/MPL เป็น freeware ที่ใช้งานเชิงพาณิชย์

2.2.3 Hunspell

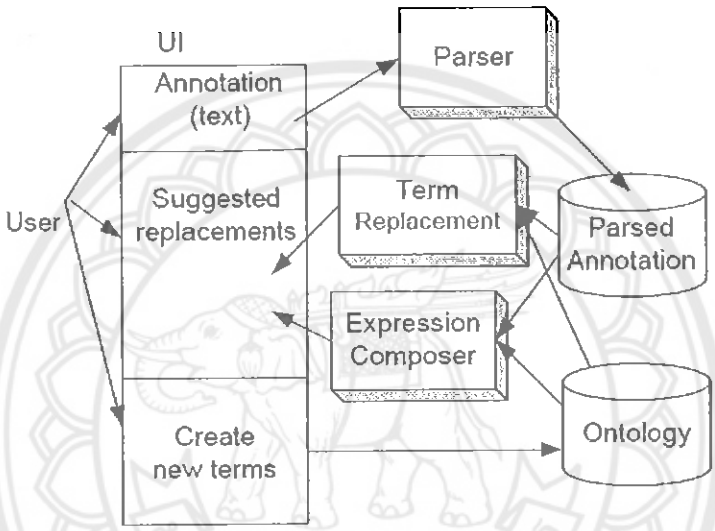
Hunspell เป็นส่วนของการเช็คความซับซ้อนของคำ โดยเน้นถึงรากศัพท์เดิม

2.2.4 MyThes

MyThes คือ ตัวที่หา Synonym จะหาคำที่ที่แตกต่างกันที่มีความหมายเหมือนกันออกมาเพื่อให้ผู้ใช้มีตัวเลือกสำหรับการค้นหาคำมากขึ้น

2.3 ศึกษาการทำงานของส่วนที่ติดต่อกับผู้ใช้ (User Interface)

จากการศึกษาตัวอย่างการทำงานของ Autonomous Content Entity (ACE) architecture [31]



รูปที่ 2.2 แสดงตัวอย่างการทำงานของ ACE architecture [31]

จากรูปเป็นการค้นหา User Interface จากออนโทโลยี ซึ่งมี 4 ขั้นตอน ดังนี้

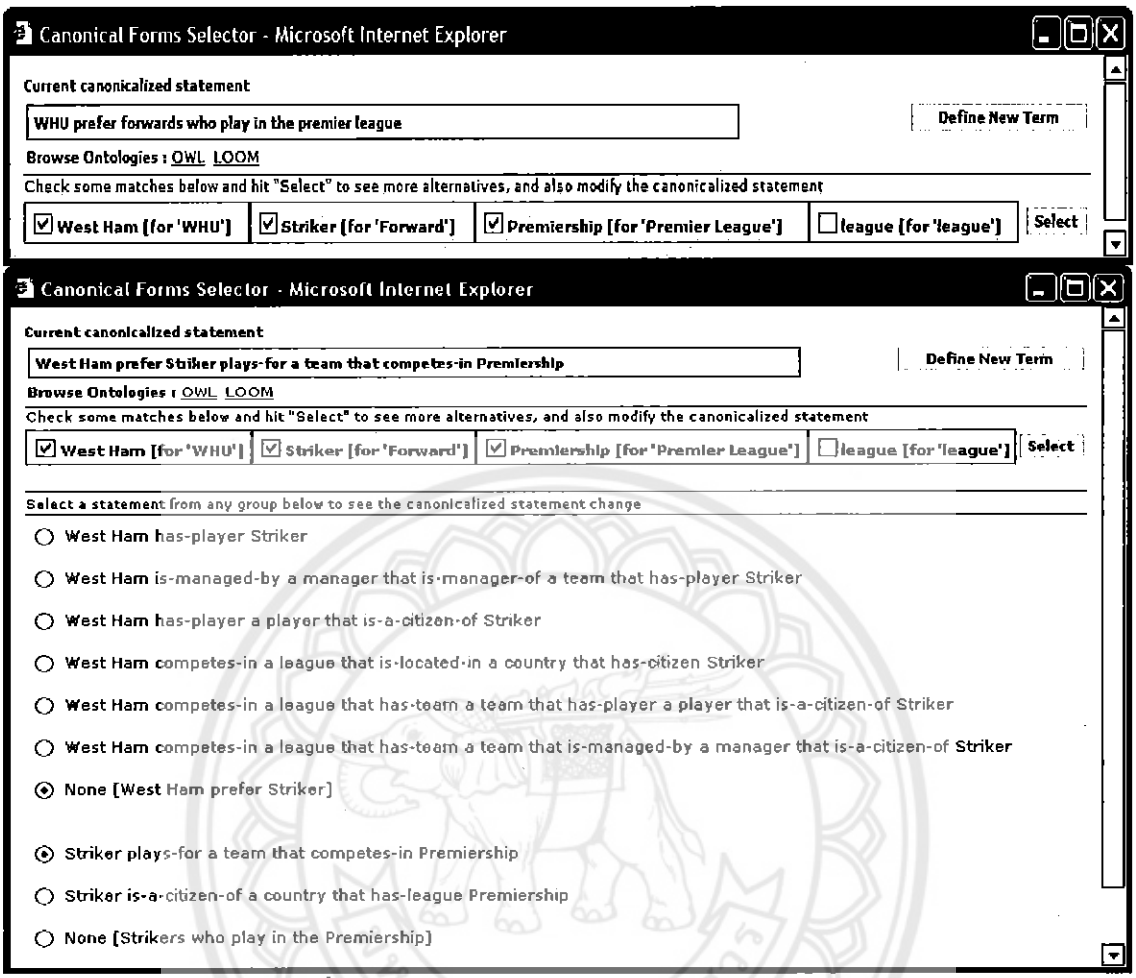
2.3.1 Term Replacement คือการหาคำมาแทนที่คำที่ผู้ใช้กรอกเข้ามา จากในตัวอย่าง ได้แนะนำให้แทนที่คำว่า "forwards" ด้วยคำว่า "strikers" ซึ่งการหาคำดังกล่าวใช้การหาคำที่มีความหมายเหมือนกัน (Synonym) โดยใช้ wordNet

2.3.2 Parser จะทำการดึงข้อมูลที่ถูกอธิบายด้วย Web Ontology Language (OWL)

2.3.3 Expression Composer นำข้อมูลที่ดึงจาก Parser มาค้นหาความสัมพันธ์ที่เป็นไปได้และส่วนที่สามารถเชื่อมโยงกับออนโทโลยี

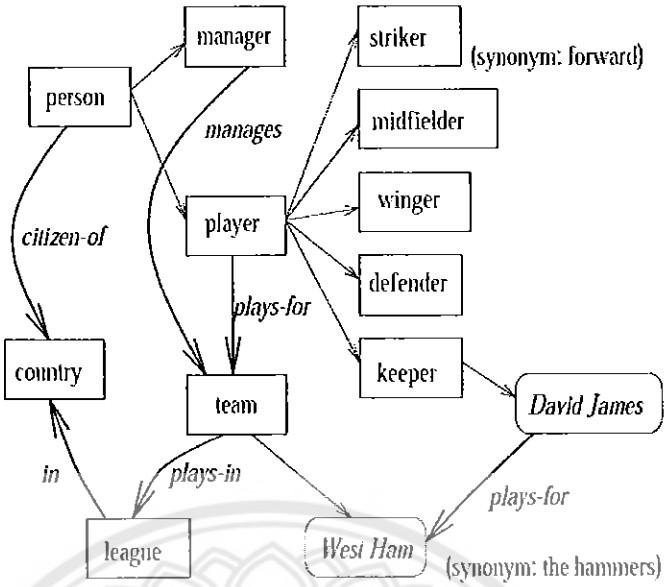
2.3.4 Create new terms ผู้ใช้สามารถสร้างประโยคขึ้นใหม่จากออนโทโลยีที่มีอยู่ โดยผู้ใช้เลือกคำที่ต้องการใช้ในการสร้างประโยคใหม่ได้ด้วยตัวเอง

จากตัวอย่างการทำงานดังที่กล่าวมาข้างต้น แสดงดังรูปต่อไปนี้



รูปที่ 2.3 แสดงตัวอย่างระบบ query refinement [31]

จากการศึกษาตัวอย่างความสัมพันธ์ของโหนดต่างๆในออนโทโลยี เพื่อนำมาประยุกต์ใช้ในการหาความสัมพันธ์ของโหนดแต่ละโหนดบนออนโทโลยี สำหรับการสร้างระบบค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ



รูปที่ 2.4 แสดงความสัมพันธ์ของโหนดต่างๆ ในออนโทโลยี [31]

จากรูปที่ 2.4 แสดงความสัมพันธ์ของโหนดต่างๆ ในออนโทโลยี ซึ่งเชื่อมโยงความสัมพันธ์ของแต่ละโหนด รวมทั้งโหนดที่สร้างขึ้นใหม่จากคำที่มีความหมายเหมือนกัน (Synonym) ด้วย

2.4 ศึกษาการทำงานของ Resource Description Framework (RDF)

Resource Description Framework (RDF) คือการกำหนดกรอบให้กับระบบเพื่ออธิบายข้อมูล ประกอบด้วย 3 ส่วน คือ Resource, Property และ คำที่มาอธิบาย Property

Resource คือส่วนที่ข้อมูลที่ถูกนำมาอธิบาย เช่น ชื่อเว็บเพจ วันที่ปรับปรุงเว็บเพจ ผู้เขียนเว็บเพจ เป็นต้น Property คือส่วนที่อธิบายความสัมพันธ์ของ resource ส่วนการอธิบายคุณลักษณะของ resource ด้วยชื่อของ Property กับค่าของ Property ของ resource นั้นจะเรียกว่า literal นั่นก็คือข้อความที่เราเขียนลงใน source code นั่นเอง

คุณสมบัติของ RDF มีหลายอย่าง ได้แก่ ช่วยบอกรูปแบบข้อมูลไวยากรณ์ของแต่ละส่วนที่ใช้แลกเปลี่ยนข้อมูลกัน ซึ่งจะเขียนด้วยภาษา XML และภาษานี้เมื่อนำมาใช้โดย RDF จะเรียกว่า RDF/XML ซึ่งสิ่งที่ได้จาก RDF/XML นี้สามารถใช้แลกเปลี่ยนระหว่างคอมพิวเตอร์ต่างประเภทกันได้ นั่นคือระบบปฏิบัติการที่ต่างกันหรือใช้แอพพลิเคชันที่ใช้ภาษาต่างกันก็สามารถเข้าใจภาษานี้ได้ นอกจากนี้ RDF ยังเป็นส่วนหนึ่งของ W3C's Semantic Web Activity ที่ได้รับการสนับสนุนจาก W3C

2.5 ศึกษาโครงสร้างข้อมูล North American Industry Classification System (NAICS)

North American Industry Classification System (NAICS) เป็นการจัดหมวดหมู่ของข้อมูลในเว็บไซต์ มันจะแบ่งประเภทสิ่งต่างๆออกมา โดยแต่ละแบบจะมีรหัส เพื่อให้ง่ายต่อการค้นหาข้อมูล ทำให้เข้าถึงข้อมูลเหล่านั้นได้อย่างรวดเร็ว

ตารางที่ 2.1 ตัวอย่างลำดับชั้นของ NAICS [10]

NAICS level	Example #1		Example #2	
	NAICS code	Description	NAICS code	Description
Sector	31-33	Manufacturing	51	Information
Subsector	334	Computer and electronic product manufacturing	513	Broadcasting and telecommunications
Industry group	3346	Manufacturing and reproduction of magnetic and optical media	5133	Telecommunications
Industry	33461	Manufacturing and reproduction of magnetic and optical media	51332	Wireless telecommunications carriers, except satellite
U.S. Industry	334611	Reproduction of software	513321	Paging

2.6 ศึกษาการนับคำ (part-of-speed tagger)

2.6.1 ข้อมูลทั่วไปเกี่ยวกับ Stanford Log-linear Part-Of-Speech Tagger

Part-Of-Speech Tagger (POS Tagger) คือ software สำหรับอ่านไฟล์ข้อมูล และทำการแยกประเภทของคำ เช่น noun , verb , adjective เป็นต้น แม้ว่าจะมีโปรแกรมการคำนวณการทำ tag POS เช่น "noun-plural" Software ซึ่งยังนำมาประยุกต์ใช้ด้วย Java ในการแยกประเภทของคำ ซึ่งมีบทความที่อธิบายไว้ดังต่อไปนี้

บทความเรื่อง Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger เขียนขึ้นโดย Kristina Toutanova และ Christopher D.Manning เป็นผู้เขียนขึ้น เมื่อปี 2000 เป็นเอกสารทางวิชาการอยู่ใน Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000) หน้า 63-70

Kristina Toutanova , Dan Klein, Christopher Manning และ Yoram Singer เป็นผู้เขียนขึ้นในปี 2003 ที่มีเนื้อหาเกี่ยวข้องกับเรื่อง Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network เอกสารทางวิชาการจะอยู่ในเรื่อง HLT-NAACL 2003 หน้า 252 -259

Tagger ที่เขียนโดย Kristina Toutanova ในครั้งนั้นโดยมี Dan Klein, Christopher Manning, William Morgan, Anna Rafferty, and Michel Galley ผู้ร่วมทำการปรับปรุงในเรื่องความเร็ว ประสิทธิภาพการใช้งาน และการสนับสนุนในภาษาอื่นด้วย

ระบบต้องติดตั้ง Java 1.5 ขึ้นไป ซึ่งขึ้นอยู่กับว่าจะใช้ java 32 bit หรือ 64 bit ด้วย และความซับซ้อนของรูปแบบ tag ที่เกิดขึ้นเกี่ยวกับหน่วยความจำ (memory) คือว่าต้องใช้ memory ตั้งแต่ 60-200 MB เพื่อให้สามารถใช้ซอฟต์แวร์นี้ได้ (จำเป็นต้องใส่ค่า java option ก่อน เช่น java – mx 200m)

Part-of-speech tagger ในรูปแบบภาษาอังกฤษจะแสดงประเภทของคำโดยยึดตาม Penn Treebank tag set ซึ่งรายละเอียดจะแสดงไว้ในภาคผนวก

2.6.2 ข้อมูลขาเข้าและข้อมูลขาออกของ Stanford Log-linear Part-Of-Speech Tagger

โปรแกรม Stanford Log-linear Part-Of-Speech สามารถรับข้อมูลขาเข้าเป็นไฟล์ประภทต่างๆ ได้มากมาก อาทิ เช่น text ,xml, tsv , doc ,dox, dotx และ dotm file เป็นต้น และหลังจากการทำงานของโปรแกรมจะได้ข้อมูลขาออกเป็นคำแต่ละคำในไฟล์ขาเข้าและประเภทของคำดังกล่าว

```
D:\POS tagger>java -mx300m -classpath stanford-postagger.jar edu.stanford.nlp.ta
gger.maxent.MaxentTagger -model models/left3words-wsj-0-18.tagger -textFile samp
le-input.txt > sample-output.txt
Loading default properties from trained tagger models/left3words-wsj-0-18.tagger

Reading POS tagger model from models/left3words-wsj-0-18.tagger ... done [1.3 se
c].
Tagged 72 words at 782.61 words per second.
```

(a)

A passenger plane has crashed shortly after take-off from Kyrgyzstan's capital, Bishkek, killing a large number of those on board. The head of Kyrgyzstan's civil aviation authority said that out of about 90 passengers and crew, only about 20 people have survived. The Itek Air Boeing 737 took off bound for Mashhad, in north-eastern Iran, but turned round some 10 minutes later.

(b)

A_DT passenger_NN plane_NN has_VBZ crashed_VBN shortly_RB after_IN take-off_NN from_IN Kyrgyzstan_NNP 's_POS capital_NN ,_, Bishkek_NNP ,_, killing_VBG a_DT large_JJ number_NN of_IN those_DT on_IN board_NN ._.
The_DT head_NN of_IN Kyrgyzstan_NNP 's_POS civil_JJ aviation_NN authority_NN said_VBD that_IN out_IN of_IN about_IN 90_CD passengers_NNS and_CC crew_NN ,_, only_RB about_IN 20_CD people_NNS have_VBP survived_VBN ._.

(c)

รูปที่ 2.5 a) แสดงตัวอย่างการรัน Part-Of-Speech Tagger ผ่าน command line โดยมีข้อมูลขาเข้าเป็น text file
b) แสดงตัวอย่างข้อมูลขาเข้าที่เป็น text file
c) แสดงตัวอย่างข้อมูลขาออกที่เป็น text file

```
D:\POS tagger>java -mx300m -classpath stanford-postagger.jar edu.stanford.nlp.ta
gger.maxent.MaxentTagger -model models/left3words-wsj-0-18.tagger -textFile my_x
ml.xml >my_xml-output.xml
Loading default properties from trained tagger models/left3words-wsj-0-18.tagger

Reading POS tagger model from models/left3words-wsj-0-18.tagger ... done [1.5 se
c].
Tagged 338 words at 149.43 words per second.

D:\POS tagger>
```

```
<?xml version="1.0" encoding="UTF-8"?>

  <entry>

   <title>kplawver: Walking the DOM like Huggybear.</title>

   <content type="html">kplawver: Walking the DOM like Huggybear.</content>

   <id>tag:twitter.com,2007:http://twitter.com/kplawver/statuses/18699620902</id>

   <published>2010-07-16T16:40:34+00:00</published>

   <updated>2010-07-16T16:40:34+00:00</updated>

   <link type="text/html" rel="alternate" href="http://twitter.com/kplawver/statuses/18699620902"/>

   <link type="image/jpeg" rel="image"
href="http://a3.twimg.com/profile_images/980477983/oh_lawver_normal.jpg"/>

   <author>

    <name>Kevin Lawver</name>

    <uri>http://lawver.net</uri>
```

```
-LRB-_-LRB- ?_
xml_NN version_NN =_SYM ``_`` 1.0_CD "_" encoding_VBG =_SYM ``_`` UTF-8_NN "_" ?_-RRB-_-
RRB-
<entry>_NNP <title>_NNP kplawver_NN :_: Walking_VBG the_DT DOM_NNP like_IN Huggybear_NNP
._.
</title>_NNP <content type="html">_NNP kplawver_NN :_: Walking_VBG the_DT DOM_NNP like_IN
Huggybear_NNP ._.
</content>_NNP <id>_NNP tag_NN :_: twitter.com_VB ,2007_CD :_:
http://twitter.com/kplawver/statuses/18699620902_NNS </id>_VBP <published>_JJ 2010-07-16T16_NNP
:40:34_CD +00:00_CD </published>_NNP <updated>_NNP 2010-07-16T16_NNP :40:34_CD +00:00_CD
</updated>_NN <link type="text/html" rel="alternate"
href="http://twitter.com/kplawver/statuses/18699620902"/>_CD <link type="image/jpeg" rel="image"
href="http://a3.twimg.com/profile_images/980477983/oh_lawver_normal.jpg"/>_CD <author>_NNP
```

รูปที่ 2.6 a) แสดงตัวอย่างการรัน Part-Of-Speech Tagger ผ่าน command line โดยมีข้อมูลขาเข้าเป็น xml file

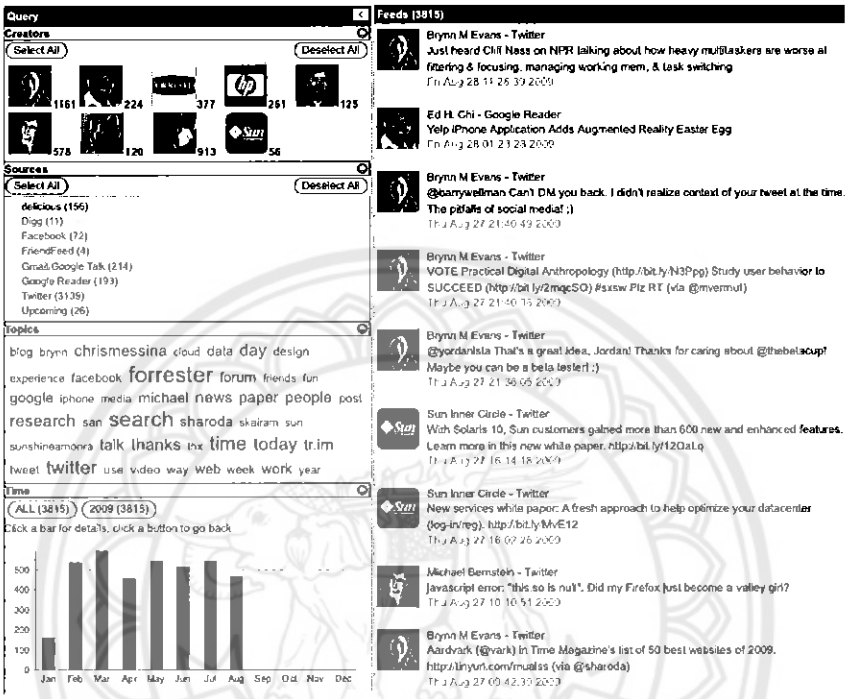
b) แสดงตัวอย่างข้อมูลขาเข้าที่เป็น xml file

c) แสดงตัวอย่างข้อมูลขาออกที่เป็น xml file

2.7 ศึกษาการสร้าง tag cloud

2.7.1 ศึกษาการทำงานของ FeedWinnower

จากการศึกษาตัวอย่างการทำงานของ FeedWinnower เพื่อนำมาแนวคิดแล้วประยุกต์ใช้ ในการ สร้างระบบการค้นหาและแสดงเอกสารทางวิชาการ โดยใช้คุณลักษณะต่างๆของเอกสาร



รูปที่ 2.7 แสดงตัวอย่าง user interface ของ FeedWinnower ซึ่งประกอบไปด้วย facet control ทางฝั่งซ้ายของรูป และ feed ทางฝั่งขวาของรูป

เราจะนำความรู้เรื่อง Topic Facet มาใช้เป็น แนวคิดในการสร้าง tag cloud ของเอกสารทางวิชาการ และความรู้เรื่อง Time Facet มาใช้เป็นแนวคิด ในการสร้าง time line ของเอกสารทางวิชาการ ซึ่งการสร้าง Facet แบ่งออกเป็น 4 แบบ ตามการใช้งาน

2.7.1.1) Topic Facet

ในทางปฏิบัติ สำหรับ feed reader ที่มีอยู่ของผู้ใช้ จะเริ่มต้นจากการ list feed ที่ยังไม่ได้อ่าน เพื่อเอาไปสร้างแบบจำลองของหัวข้อที่อภิปราย เป้าหมายของ topic facet คือ การให้ข้อมูลระดับสูง และรวดเร็ว บางหัวข้ออาจจะเก่าแล้ว แต่ถูกใช้งานบ่อย อาจจะถูกแยกออกมา

Feed ทั่วๆไปมีตัวอย่างข้อความสั้นๆ เช่น Tweet เป็นการดึงหัวข้อจากข้อความกลุ่มหนึ่ง โดยใช้หลายวิธีและใช้เทคนิคในการแยกแยะคำนาม สำหรับในอดีตนั้นจะใช้นามวลี แสดงใจความหลักของข้อความตัวอย่าง ก่อนอื่นต้องทำการแยกแยะคำนามที่ปรากฏในหัวข้อ โดยใช้โปรแกรม Stanford part-of speech tagger (nlp.stanford.edu/software/tagger.shtml) จากนั้นเมื่อแยก

คำนามได้แล้ว ให้ทำการนับจำนวนคำนามเหล่านั้นให้มีค่าเป็น n โดยคำนามที่พบสูงสุดจำนวน n ครั้ง ซึ่งจะถูกเลือก และปรากฏบน tag cloud

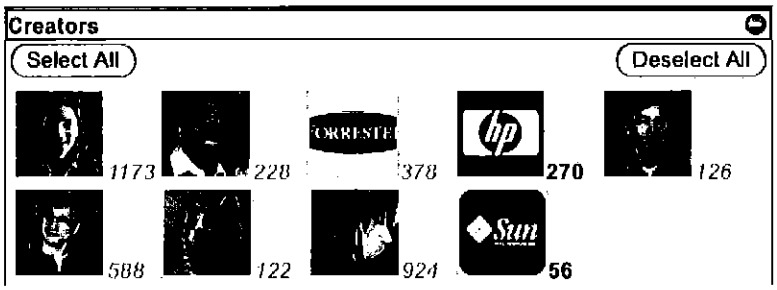
รูปที่ 2.8 ตัวอย่างของTopic Facet แสดง subset ที่เกี่ยวข้องกับ iphone ที่ปรากฏบน tag cloud

ผู้ใช้สามารถคลิกเข้าไปดูหัวข้อที่อยู่ในหน้า tag cloud ได้ เช่น กดเข้าไปในแท็กที่ชื่อว่า iphone ก็จะมีเซตย่อย (subset) ที่เกียวข้องกับ iphone ขึ้นมา ใน tag cloud ที่อัพเดตข้อมูลที่เราต้องการ (ดังรูปที่ 2.8) โดย tag cloud ที่ปรับปรุงใหม่นี้ได้มาจากการทำตาม algorithm ที่ได้อธิบายมาแล้วข้างบน การเลือกหัวข้อเพิ่มจากแท็กที่มีอยู่จะทำให้ลดจำนวนหัวข้อที่ยังเหลืออยู่ tag cloud และจะแสดงข้อมูลเกี่ยวกับหัวข้อทุกอันที่เราเลือกไปทำงานโดยใช้ AND query ผู้ใช้เองสามารถทำซ้ำได้ว่าจะเลือกหรือ ไม่เลือกหัวข้อได้

เทคนิคการแยกแยะคำนาม ในการสร้าง Topic Facet ถือว่าเป็นวิธีที่ดีที่สุด ส่วนวิธีอื่นอย่างเช่น TF-IDF และ term expansion algorithm แต่เมื่อเราใช้วิธีพวกนี้ปรากฏว่าไม่ได้ผลตามที่คาดหมาย นอกจากนั้น การเน้นคำนามหรือวลีในรายการที่ยังเหลืออยู่จะช่วยอธิบายให้ผู้ใช้ได้ทราบเหตุผลว่า เพราะเหตุใดหัวข้อนี้ถึงถูกจัดอยู่ในผลการค้นหา

2.7.1.2) People Facets

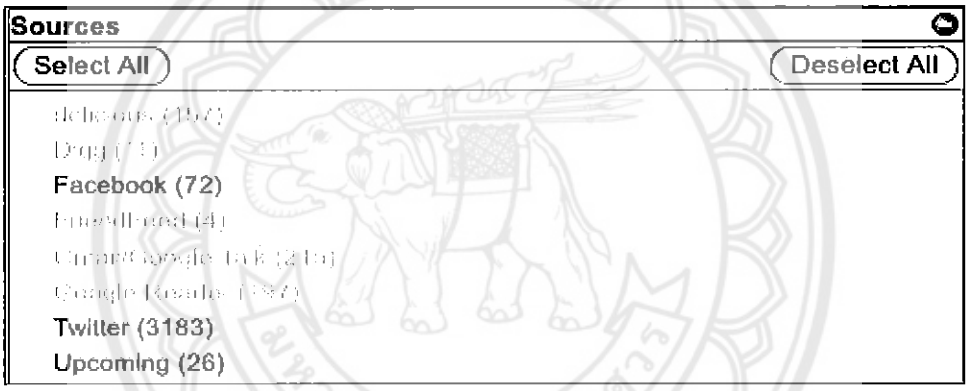
ในส่วนของ People เป็นส่วนที่อธิบายว่า ใครสร้างหัวข้อเหล่านี้ขึ้นมา ไอคอนรูปภาพจะแสดงรูปภาพของผู้สร้าง (อาจจะเป็นบุคคลหรือองค์กรก็ได้) และจำนวนหัวข้อที่บุคคลนี้สร้าง ถ้าเราคลิกเข้าไปจะพบกับเซตย่อย (subset) ของบุคคลเหล่านี้ เช่น Hp และ Sun ซึ่งก็จะกรองข้อมูลโดยเอาแต่ผลิตภัณฑ์ของ Hp และ Sun



รูปที่ 2.9 ตัวอย่าง people facet แสดงชื่อของผู้เขียนและจำนวนบทความของผู้เขียนคน

2.7.1.3) Source Facets

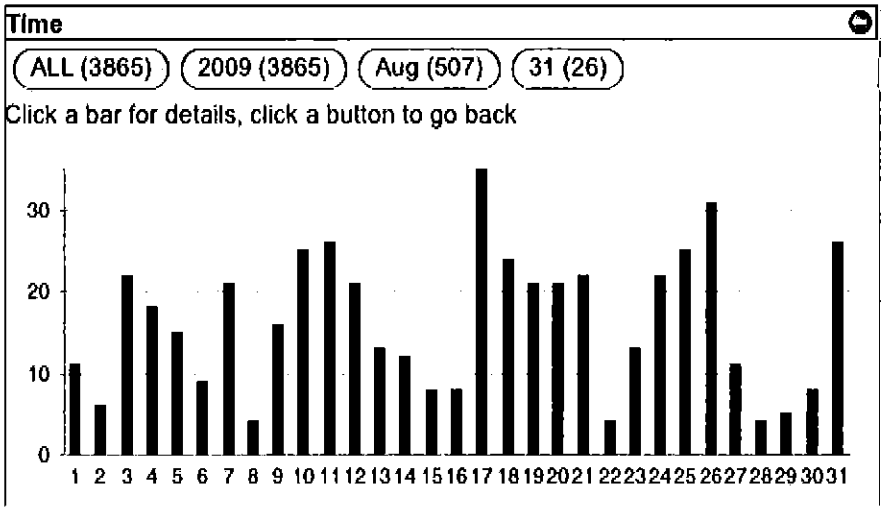
ในส่วนของ Source เป็นส่วนที่อธิบายว่า ข้อมูลนี้มาจากไหน source facet แสดงในรูปที่ 2.10 มีรายการของแหล่ง ให้บริการ โดยมีการแสดงว่าแหล่งที่มานี้มีจำนวนบทความเท่าไร ผู้ใช้สามารถคลิกเลือกแต่ละแหล่งที่มาเพื่อเข้าไปดูบทความของแหล่งที่มานั้นได้



รูปที่ 2.10 ตัวอย่างsource facet แสดงรายการของแหล่งที่มาและจำนวนบทความจากแหล่งที่มาดังกล่าว

2.7.1.4) Time Facets

ในส่วนของ Time เป็นส่วนที่อธิบายว่า ข้อมูล แต่ละแถบแสดงถึงจำนวนบทความที่ถูกสร้างในช่วงเวลานั้นๆ อาจจะแบ่งเป็น วัน เดือน ปี โดยถ้าเราคลิกที่แถบ ผู้ใช้สามารถมองลึกลงไปถึงรายละเอียด (เช่น ถ้าเราดูแบบเป็นเดือน เราก็คลิกลงไปดูรายละเอียดของแต่ละวันได้) และมีตัวเลือกด้านบน เพื่อที่จะกลับไปสู่มุมมองหลัก (กลับไป ดูเป็นรายปี เมื่อเรากำลังดูเป็นรายเดือนอยู่)



รูปที่ 2.11 ตัวอย่าง time facet แสดงจำนวนบทความในแต่ละวันของเดือน

2.7.2 การแท็ก (Tagging) [3]

การแท็ก (Tagging) เป็นกระบวนการเพิ่มข้อความที่มีรูปแบบอิสระ คำหรือวลีโดยคำพวกนี้จะมี ความเกี่ยวข้องกับผู้ใช้ รูปภาพ บทความ สินค้า วิดีโอ หรืออื่นๆ แท็ก (tag) สามารถแสดงใช้แสดงตัว เชื่อมโยงนำทางแบบไดนามิก (dynamic navigation links

แนวทางในการสร้างแท็ก มีดังต่อไปนี้

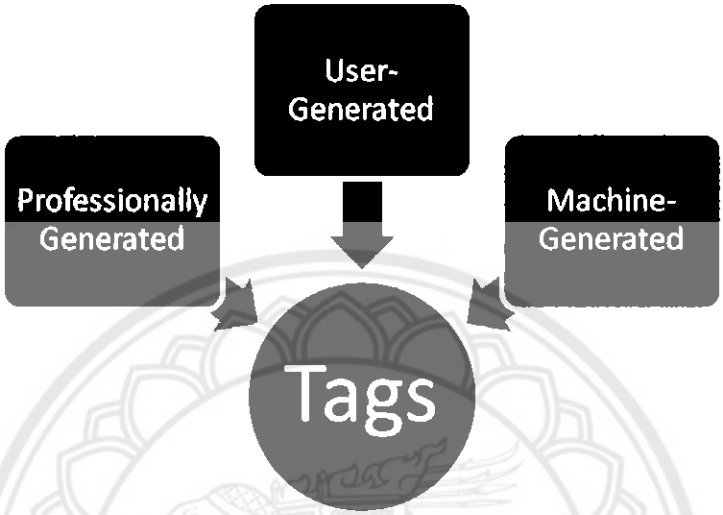
- ควรสร้าง tag dictionary สำหรับในอนาคต tag ของเราจะมีความสัมพันธ์แบบ "is-a"
- อาจ จะสร้าง synonyms dictionary ซึ่งรวมกลุ่มของคำที่มีความหมายเหมือนกัน
- หลีกเลี่ยงแท็กที่มีความหมายกำกวม โดยใช้วลีที่ทำให้เลือกได้อย่างเจาะจงกว่า
  - อย่าใช้แท็กหลายตัวที่สื่อความหมายเดียวกัน ก็เพื่อที่จะแสดงว่าเป็นเอกพจน์ หรือ พหุพจน์ คล้าย ๆ กับในส่วนของตัวอักษร ที่ไม่ต้องสนใจว่าจะเป็นอักษรตัวใหญ่หรือ ตัวเล็ก โดยปรับให้เป็นตัวเล็กเหมือนกันให้หมด
  - การหาวลีที่ประกอบด้วยหลายคำมาเป็นแท็กนั้น ใช้ automated algorithm และอาจจำ เป็นต้องใช้ phrase dictionary
- เราสามารถใช้ tag ทั้ง 3 ประเภทรวมกันได้

หน้าที่ของแท็ก (tag) มีดังนี้

- สร้างแบบจำลองของเมตาดาตา (ในรูปแบบของ vector)
- สร้างตัวเชื่อมโยงนำทางแบบไดนามิก (dynamic navigation links) ของตัวแอพพลิเคชั่น (application) ตัวอย่างเช่น tag cloud บนเว็ปไซต์
- ใช้เมตาดาต้าเพื่อปรับแต่งและการเชื่อมต่อระหว่างผู้ใช้หนึ่งกับผู้ใช้อื่น ๆ

- รายการที่คั่นไว้ (bookmark) จะสามารถใช้ร่วมกับผู้ใช้คนอื่นได้

ประเภทของแท็ก (tag) ถูกแบ่งออกเป็น 3 แบบหลักๆ คือ Professionally generated Tags , User-generated Tags และ Machine-generated Tags



รูปที่ 2.12 แสดงวิธีในการสร้าง tag cloud ทั้ง 3 แบบ

1) Professionally generated Tags

ในปัจจุบันข้อมูลมีรูปแบบที่หลากหลาย มีเนื้อหาที่น่าสนใจและมีการแสดงรูปแบบของข้อมูล ที่แตกต่างกันออกไป เช่น เนื้อหา, วิดีโอ, ภาพถ่าย, บล็อก เป็นต้น ซึ่งแท็กชนิดแรกนี้ คลอบคลุมการ สร้างแท็ก โดย domain expert โดย Professionally generated Tags ต้องมีคุณลักษณะสำคัญดังต่อไปนี้ดึงแนวคิดหลักของเนื้อความนั้นๆออกมา

- กำหนดความหมายโดยใช้คำที่ไม่ได้อยู่ในบทความ
- สามารถแสดงข้อมูลให้ผู้ใช้ได้
- ดึงส่วนที่อาจจะไม่ใช่แนวคิดหลักแต่เป็นเนื้อหาที่ความน่าสนใจออกมาแสดงได้
- รู้จักและแยกแยะคำที่มีความหมายเหมือนกันได้
- สามารถเป็นวลีที่มีหลายคำได้
- กลุ่มของคำพวกนี้สามารถควบคุมได้

2) User-generated Tags

การให้ผู้ใช้สร้าง tag cloud ด้วยตัวเอง การกำหนดแท็กเกิดจากผู้ใช้เอง ว่าจะให้ความสำคัญกับ คำใดมากกกว่ากัน โดย User-generated Tags ต้องมีคุณลักษณะสำคัญดังต่อไปนี้

- ดึงแนวคิดหลักของเนื้อความนั้นๆออกมา
- ดึงส่วนที่อาจจะไม่ใช่แนวคิดหลักแต่เป็นเนื้อหาที่ความน่าสนใจออกมาแสดงได้
- สามารถเป็นวลีที่มีหลายคำได้
- ประกอบไปด้วยคำที่หลากหลายที่สื่อความหมายได้ดี
- ข้อมูลมีความเกี่ยวข้องระหว่างผู้ใช้และรายการของข้อมูล

3) Machine-generated Tags

แท็กประเภทนี้จะถูกสร้างโดยใช้ automated algorithm ซึ่งทำงานโดยอัตโนมัติ ซึ่ง algorithm นี้ จะสร้างแท็กโดยการแยกวิเคราะห์ (parsing) เนื้อความและหาคำหรือวลีที่มีความเหมาะสมที่จะเป็นแท็ก

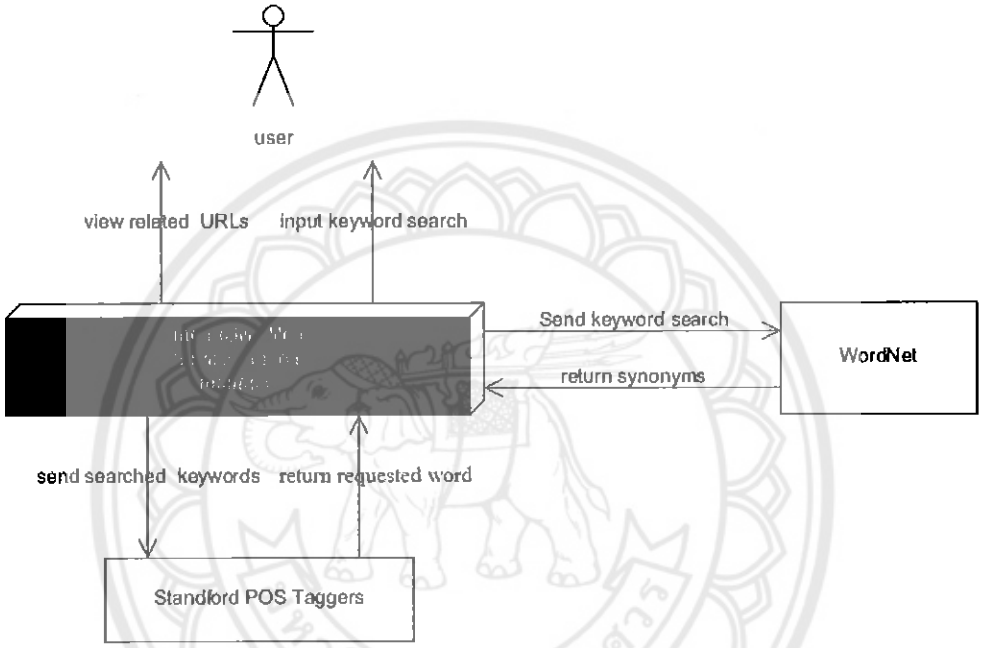
- ใช้คำที่มีอยู่ในข้อความนั้น โดยมีข้อยกเว้นของคำ ที่เป็นคำเหมือน
- ส่วนใหญ่จะใช้คำเดี่ยว เพราะวลีที่มีหลายคำนั้นยากที่จะดึงออกมาอาจจะเกิดแท็กที่ไม่ดี เช่น แท็กที่มีความหมายกำกวมทั้งที่ขึ้นอยู่กับบริบทและไม่ขึ้น อยู่กับบริบท เช่น คำว่า gain มีได้หลายความหมาย ทั้ง height gain, weight gain, stock price gain, capital gain, amplifier gain และอื่นๆวิธีแก้ปัญหานี้ก็คือการตรวจหาวลีที่ทำ ให้เลือกได้อย่างเจาะจงกว่า

ซึ่งแท็กที่ถูกใช้ในระบบที่พัฒนาเป็นแท็กที่กำหนดโดยผู้ใช้ โดยข้อมูลการแท็กจะถูกจัดการด้วยระบบฐานข้อมูลแท็ก

บทที่ 3
ขั้นตอนและวิธีการดำเนินงาน

3.1 การออกแบบระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ

3.1.1 Conceptual design



รูปที่ 3.1 Conceptual design for Interactive Web Service Search Interface

จากรูปที่ 3.1 ระบบที่พัฒนาขึ้นมีองค์ประกอบที่เกี่ยวข้องดังต่อไปนี้

- ผู้ใช้ระบบ (Client) คือ ผู้ใช้ที่ให้ข้อมูลอินพุตและตอบสนองแก่ระบบ เพื่อค้นหาข้อมูล

- WordNet คือ ส่วนของแอพพลิเคชั่นที่นำมาใช้ในการค้นหาคำที่มีความหมายเหมือนกัน (Synonym)

- Stanford POS tagger คือ ส่วนของแอพพลิเคชั่นที่นำมาใช้ในแยกประเภทของคำและเลือกเฉพาะคำนามจากประโยคค้นหา

3.1.2 โครงสร้างการทำงานของระบบและการเชื่อมต่อการทำงาน

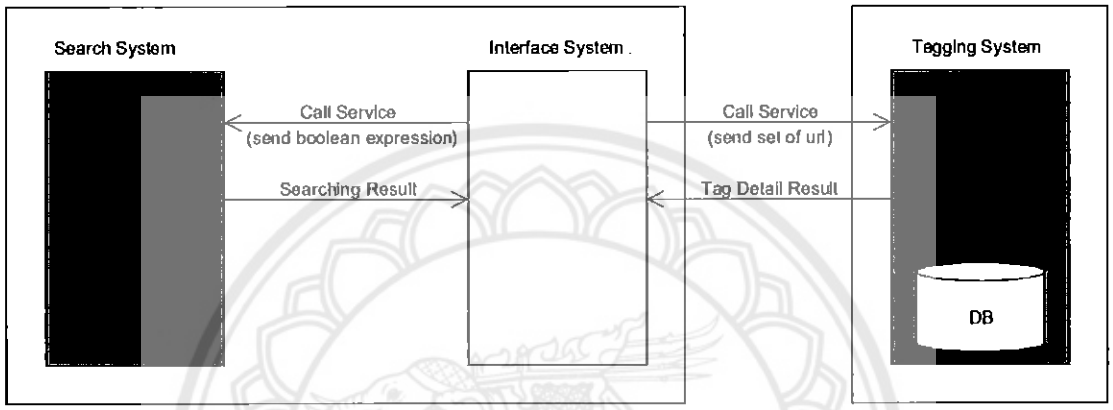
นอกจากนี้ระบบต้องทำงานเชื่อมต่อกับระบบอื่น ดังต่อไปนี้

3.1.2.1 ระบบการค้นหาเว็บเซอร์วิส (Web services Search System)

ระบบการค้นหาเว็บเซอร์วิสจะรับนิพจน์ทางตรรกศาสตร์(Boolean Expression) และส่งผลของการค้นหาเว็บเซอร์วิสไปให้ระบบแสดงผล

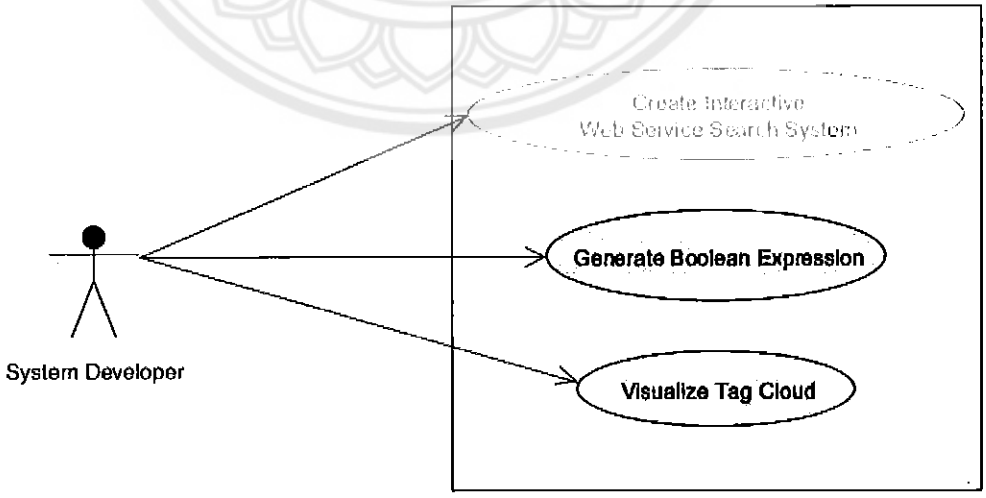
3.1.2.2 ระบบฐานข้อมูลแท็ก (Tagging System)

ระบบฐานข้อมูลแท็ก (Tagging System) จะรับเซตของเว็บเซอร์วิสในรูปแบบของ url และส่งรายละเอียดของแท็กที่เกี่ยวข้องไปให้ระบบแสดงผล



รูปที่ 3.2 แสดงโครงสร้างการทำงานของระบบและการเชื่อมต่อการทำงาน

3.1.3 Use Case Diagram



รูปที่ 3.3 Use Case Diagram in Interactive Web Service Search Interface

3.1.4 Use case description

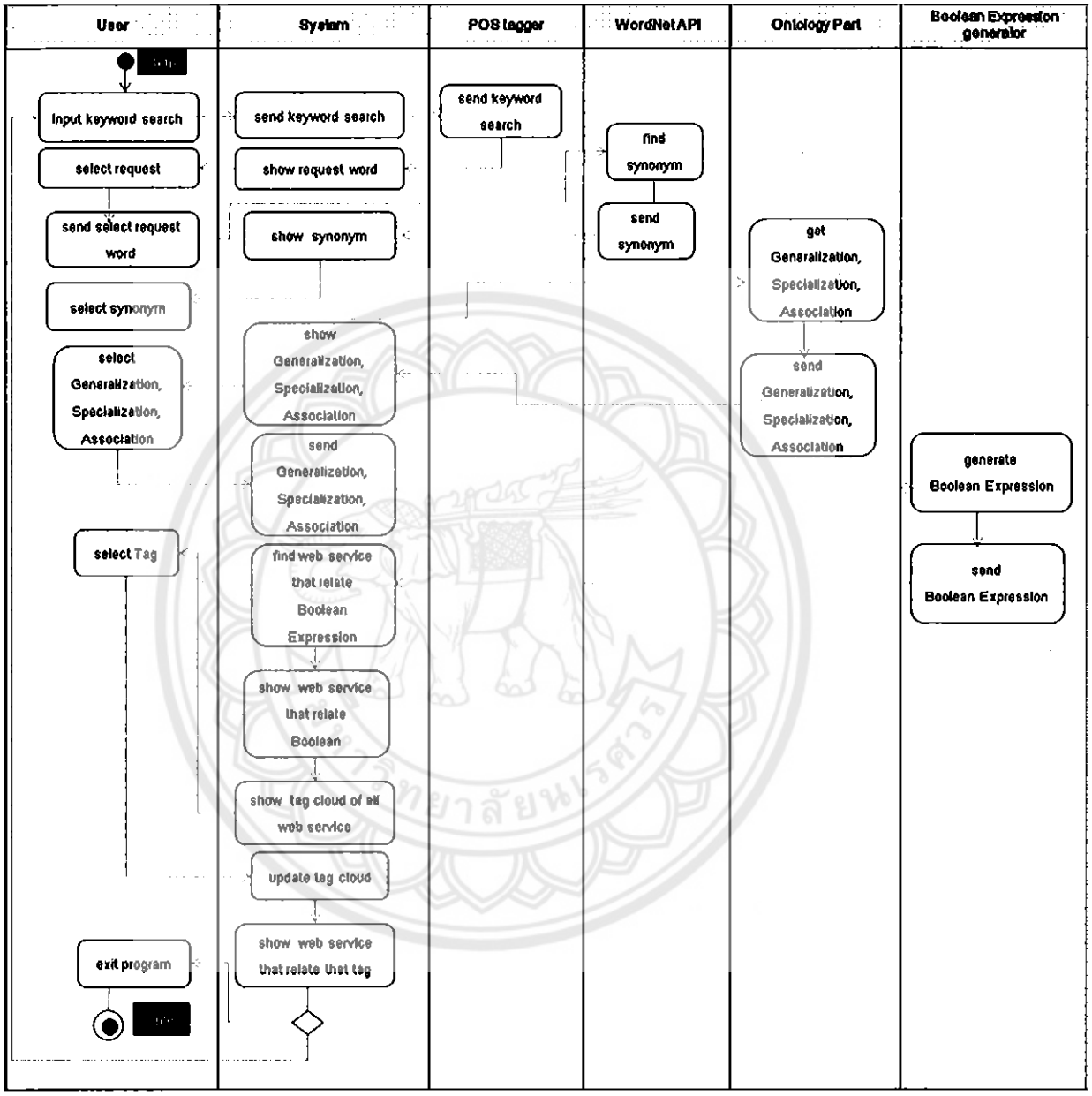
ตารางที่ 3.1 Use Case Description ของระบบค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ

ระบบค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ
1. Name : Create interactive web service search interface
2. Actor : System, user
3. Entity Condition : ผู้ใช้กรอกคำที่ต้องการค้นหา(keyword search)
4. Exit Condition : ได้เว็บเซอร์วิสที่ตรงกับความต้องการของผู้ใช้
5. Flow of Event : - ผู้ใช้กรอก keyword search - ระบบเลือกเฉพาะคำนามจาก keyword search และแสดงให้ผู้ใช้เลือกคำที่ต้องการค้นหา (request word) - ผู้ใช้เลือก request word - ระบบแสดง synonym ของ request word ที่ผู้ใช้เลือก และแสดง Generalization, Specialization และ Association ของคำๆนั้นบน ontology - ผู้ใช้เลือก synonym ที่ต้องการใช้ในการค้นหา - ระบบแสดง Generalization, Specialization และ Association ของคำๆนั้นบน synonym - ผู้ใช้เลือก Generalization, Specialization และ Association - ระบบทำการสร้าง boolean expression ที่ต้องการค้นหา - ระบบค้นหา web service ที่สัมพันธ์กับ boolean expression ที่สร้างขึ้น
6. Special Requirements : None
7. Exception : None

ตารางที่ 3.2 Use Case Description สำหรับการสร้าง tag cloud จากเว็บเซอร์วิส

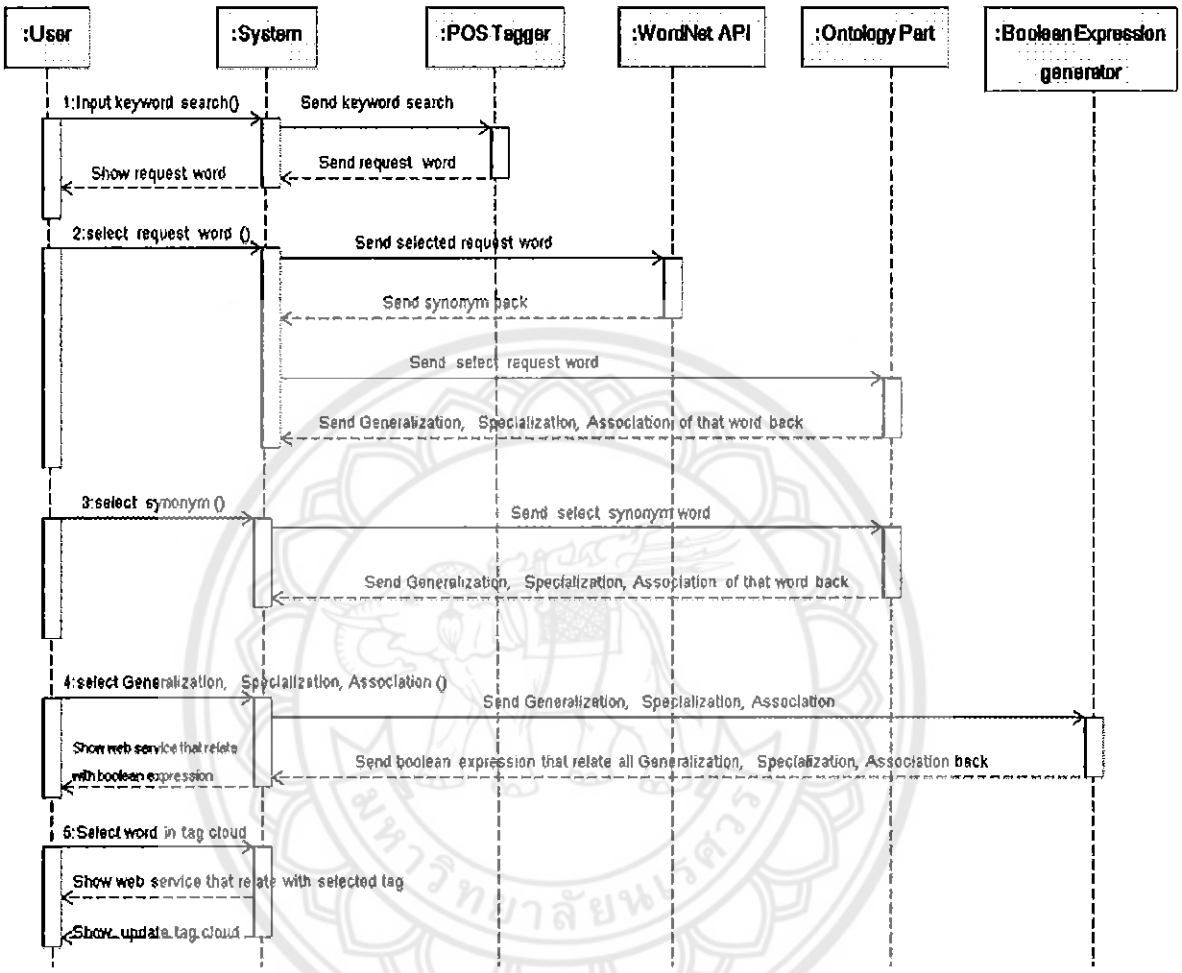
การสร้าง tag cloud จากเว็บเซอร์วิส
1. Name : Generate tag cloud from web service
2. Actor : System, user
3. Entity Condition : - ระบบค้นหา web service ที่สอดคล้องกับ boolean expression ที่ได้รับจากการ search ของผู้ใช้ - เมื่อผู้ใช้กดเลือกค่าใน tag cloud
4. Entity Condition : เมื่อผู้ใช้หยุดการค้นหา
5. Flow of Event : - ระบบนำ keyword ที่สอดคล้องกับ boolean expression ที่ได้ของ web service มาสร้าง tag cloud - หากผู้ใช้เลือกคำใน tag cloud ระบบแสดง web service ที่เกี่ยวกับคำที่เราเลือก - ระบบแสดง tag cloud ของ web service ที่เปลี่ยนแปลงหลังจากการเลือก
6. Special Requirements : None
7. Exception : None

3.1.5 Activity diagram



รูปที่ 3.4 Activity Diagram in Interactive Web Service Search Interface

3.1.6 Sequence Diagram



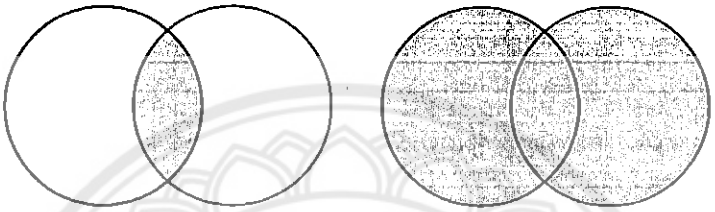
รูปที่ 3.5 Sequence Diagram in Interactive Web Service Search Interface

3.2 การสร้างนิพจน์ทางตรรกศาสตร์

ในระบบการค้นหาข้อมูลบนเว็บไซต์ จำเป็นต้องอาศัยหลักการของนิพจน์ทางตรรกศาสตร์ (Boolean expression) เพื่อหาข้อมูลความสัมพันธ์ของ URL แต่ละตัวออกมา ซึ่งมีเงื่อนไขที่แตกต่างกันตามความหมายของ Boolean expression ซึ่งการใช้งานของเงื่อนไข Boolean expression จะพิจารณาคำจากผู้ใช้เลือก โดยระบบจะวิเคราะห์ข้อมูลที่ผู้ใช้เลือกเพื่อเป็นแนวทางในการค้นหาข้อมูลให้ตรงตามความต้องการของผู้ใช้มากที่สุด

3.2.1 Boolean logic

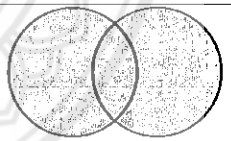
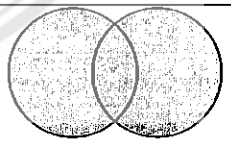
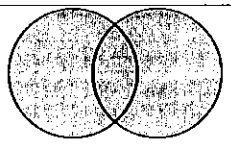
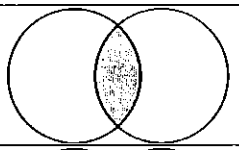
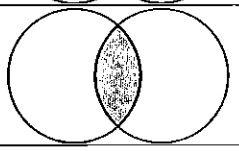
เงื่อนไข Boolean expression ที่ใช้ในการค้นหาข้อมูลมีอยู่ 2ชนิด คือ AND แทนด้วยสัญลักษณ์ ($\cup$)และ OR แทนด้วยสัญักษณ์ ($\cap$)



a. AND ($\cup$) b. OR ($\cap$)

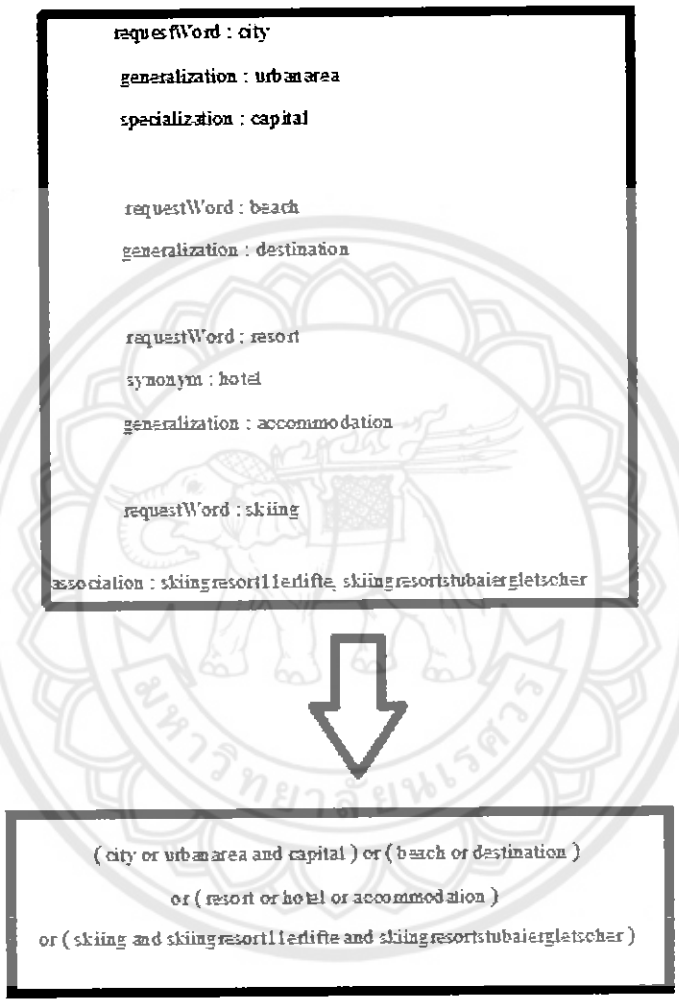
รูปที่ 3.6 แสดงเงื่อนไขของ Boolean Expression ในการดึงข้อมูล

ตารางที่ 3.3 แสดงเงื่อนไขของ Boolean logic ของข้อมูลแต่ละแบบ

Type	Boolean Logic	
Request Word	or	
Synonym	or	
Generalization	or	
Specialization	and	
Association	and	

ตัวอย่างการสร้างนิพจน์ทางตรรกศาสตร์ (Boolean expression)

ข้อมูลที่ได้จากผู้ใช้มีดังต่อไปนี้



รูปที่ 3.7 แสดงตัวอย่างการสร้างนิพจน์ทางตรรกศาสตร์ (Boolean expression)

3.3 การสร้างส่วนแสดงผลในรูปแบบของ Tag Cloud

3.3.1 เงื่อนไขของโปรแกรม

โปรแกรมจะแสดงตัวอักษรตัวใหญ่ เมื่อ tag ตัวนั้นมีความถี่มาก และในทางตรงกันข้าม จะแสดงตัวอักษรตัวเล็ก เมื่อ tag ตัวนั้นมีความถี่น้อย

3.3.2 ข้อมูลเข้า (input) ของโปรแกรม

ข้อมูลเข้า (input) คือ คำและความถี่ของคำคำนั้นที่ได้จากโปรแกรมนับความถี่ของคำ

3.3.3 ข้อมูลออก (output) ของโปรแกรม

ข้อมูลออก (output) คือ tag cloud ที่มีการกำหนดขนาดของตัวอักษรของคำแต่ละคำแตกต่างกันไปตามความถี่

รูปที่ 3.8 แสดงผลของโปรแกรมควบคุมส่วนแสดงผลของ tag cloud [34]

บทที่ 4

การพัฒนาและใช้งานระบบ

จากการศึกษาค้นคว้าข้อมูลและทฤษฎีเพื่อทำการออกแบบระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ สามารถสร้างระบบค้นหาเว็บเซอร์วิส และสร้าง tag cloud จากเว็บเซอร์วิส สำหรับบทนี้เป็นการแสดงการทำงานของระบบที่ได้จากการออกแบบ

4.1 Hardware และ Software Requirement

4.1.1 Hardware Requirement

1) หน่วยประมวลผลกลาง (CPU) Intel Pentium 4 หรือสูงกว่า

2) หน่วยความจำหลัก (RAM) 512 Megabyte ขึ้นไป

4.1.2 Software Requirement

1) ระบบปฏิบัติการวินโดว์ XP หรือสูงกว่า

2) web browser

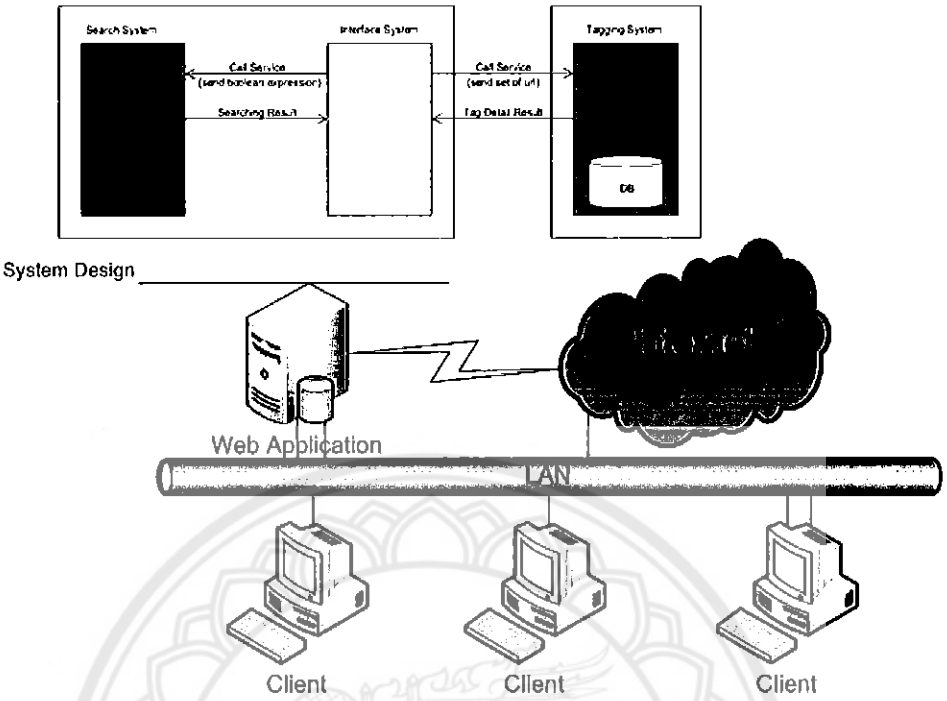
3) java version 5 ขึ้นไป

(http://www.java.com/en/download/index.jsp)

4) Microsoft visual studio 2008 professional edition ขึ้นไป

(http://www.microsoft.com)

4.2 Physical Design



รูปที่ 4.1 แสดง Physical Design ของ System

จากรูปที่ 4.1 จะเห็นว่า ผู้ใช้ (Client) สามารถใช้บริการเว็บแอพพลิเคชัน (Web Application) ผ่านอินเตอร์เน็ต โดยการทำงานภายในของเว็บแอพพลิเคชัน (Web Application) ดังกล่าวประกอบได้ด้วย 3 ระบบหลัก คือ Search System , Interface System และ Tagging System ซึ่ง Interface System เป็นระบบที่ผู้จัดทำพัฒนาขึ้น ซึ่งทำงานเป็นส่วนต่อประสานระหว่างผู้ใช้กับระบบโดยรวมและให้คำแนะนำในการค้นหา

4.3 อินพุต เอาต์พุต และเซ็ตข้อมูล

4.3.1 ข้อมูลอินพุต

4.3.1.1) อินพุตจากระบบการค้นหาเว็บเซอร์วิส(Search Engine)

อินพุตของระบบแสดงผล (Interface System) คือ เมื่อระบบได้สร้างนิพจน์ทางตรรกศาสตร์ (Boolean Expression) จากการค้นหาของผู้ใช้ นิพจน์ทางตรรกศาสตร์ ดังกล่าวจะถูกส่งไปยังส่วนค้นหาข้อมูล(Search Engine) และได้รับอินพุตเป็นผลเป็นกลุ่มของ url ที่ตรงกับเงื่อนไขในนิพจน์ทางตรรกศาสตร์ (Boolean Expression) ในรูปแบบของ xml ดังรูป 4.2

```
<?xml version="1.0" encoding="utf-8"?>
<ListURL>
  <url>http://127.0.0.1/services/1.0/1personbicycle4wheeledcar_price_service.owls</url>
  <url>http://127.0.0.1/services/1.0/1personbicyclecar_price_Kohlservice.owls</url>
  <url>http://127.0.0.1/services/1.1/author_bookrecommendedprice_service.owls</url>
  <url>http://127.0.0.1/services/1.1/4wheeledcar1personbicycle_price_service.owls</url>
  <url>http://127.0.0.1/services/1.1/3wheeledcaryear_price_service.owls</url>
  <url>http://127.0.0.1/services/1.1/_coffeeteareport_service.owls</url>
  <url>http://127.0.0.1/services/1.1/_author_CompJservice.owls</url>
  <url>http://127.0.0.1/services/1.0/book_authorbook-type_service.owls</url>
  <url>http://127.0.0.1/services/1.0/_pricecamera_Wallmartservice.owls</url>
  <url>http://127.0.0.1/services/1.0/cdplayer_recommendedprice_service.owls</url>
  <url>http://127.0.0.1/services/1.1/autobicycle_maxprice_service.owls</url>
</ListURL>
```

รูปที่ 4.2 แสดงตัวอย่างไฟล์ XML ของ URL ที่ได้จากส่วนค้นหาข้อมูล(Search Engine)

4.3.1.2) อินพุตจากระบบฐานข้อมูลแท็ก (Tagging System)

ระบบแสดงผล (Interface System) จะส่งข้อมูลกลุ่มของ url ที่ได้จากส่วนค้นหาข้อมูล(Search Engine) ไปยังระบบฐานข้อมูลแท็ก (Tagging System) และได้รับข้อมูลเกี่ยวกับแท็กต่างๆของกลุ่ม url ดังกล่าว รวมทั้งความถี่ของแท็กแต่ละแท็ก ข้อมูลดังกล่าวจะถูกนำไปแสดงผลในรูปแบบของtag cloud ต่อไป ดังรูป 4.3 ซึ่งแสดงรูปแบบข้อมูลแท็กที่ระบบได้รับ

```
<?xml version="1.0" encoding="utf-8" ?>
<ListURL>
  <url ID= ".........">
    <tag>.........</tag>
    <frequency>...</frequency>
    <tag>.........</tag>
    <frequency>...</frequency>
    <tag>.........</tag>
    <frequency>...</frequency>
  </url>

  <url ID= ".........">
    <tag>.........</tag>
    <frequency>...</frequency>
  </url>

  <url ID= ".........">
    <tag>.........</tag>
    <frequency>...</frequency>
    <tag>.........</tag>
    <frequency>...</frequency>
  </url>

</ListURL>
```

รูปที่ 4.3 แสดงตัวอย่างไฟล์ XML ของ tag ที่ได้จากระบบฐานข้อมูลแท็ก

4.3.2 ข้อมูลเอาต์พุต

4.3.2.1) เอาต์พุตที่ส่งให้แก่ส่วนค้นหาข้อมูล(Search Engine)

เอาต์พุตที่ส่งให้แก่ส่วนค้นหาข้อมูล(Search Engine) เป็นไฟล์ xml ที่ประกอบไปด้วยนิพจน์ทางตรรกศาสตร์ (Boolean Expression) และรายละเอียดเกี่ยวกับการเลือกคำหรือประโยคแนะนำของผู้ใช้ ซึ่งนิพจน์ทางตรรกศาสตร์ (Boolean Expression) จะถูกนำไปเป็นเงื่อนไขในการค้นหา ส่วนเอาต์พุตของระบบแสดงดังรูป 4.4

```
<?xml version="1.0" encoding="utf-8"?><booleanExpression
  ID="( city or urbanarea and capital ) or ( beach or destination )
       or ( resort or hotel or accommodation )
       or ( skiing and skiingresort11erlifte and skiingresortstubaiergletscher )">
  <Data>
    <requestWord>city</requestWord>
    <generalization>urbanarea</generalization>
    <specialization>capital</specialization>
  </Data>
  <Data>
    <requestWord>beach</requestWord>
    <generalization>destination</generalization>
  </Data>
```

รูปที่ 4.4 แสดงตัวอย่างไฟล์ XML ของ Boolean Expression ที่ส่งให้ส่วนค้นหาข้อมูล(Search Engine)

4.3.2.2) เอาต์พุตที่ส่งให้แก่ระบบฐานข้อมูลแท็ก

เอาต์พุตที่ส่งจากระบบการแสดงผล (Interface System) เป็นกลุ่มของurl **เพื่อร้องขอข้อมูล** แท็กจากระบบฐานข้อมูล และนำไปแสดงผลในรูปแบบของtag cloud

```
<?xml version="1.0" encoding="utf-8"?>
<ListURL>
  <url>http://127.0.0.1/services/1.0/1personbicycle4wheeledcar_price_service.owls</url>
  <url>http://127.0.0.1/services/1.0/1personbicyclecar_price_Kohlservice.owls</url>
  <url>http://127.0.0.1/services/1.0/1personbicyclecar_price_TheBestservice.owls</url>
  <url />
  <Title />
</ListURL>
```

รูปที่ 4.5 แสดงตัวอย่างไฟล์ XML ซึ่งส่งให้แก่ระบบฐานข้อมูลแท็ก (Tagging System) เพื่อร้องขอข้อมูลเกี่ยวกับแท็ก

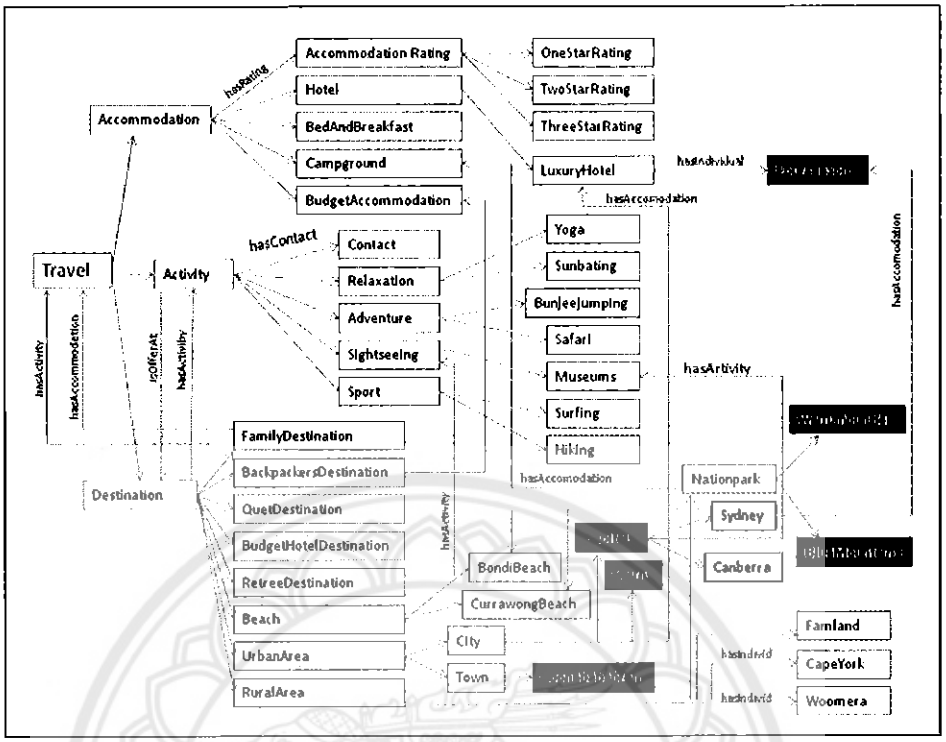
4.3.3 เซ็ตข้อมูล (Data Set)

- OWLS-TC (โดยการใช้ Test Collection ของ OWL-S และ NAICS ของ Test Collection นี้)

ที่มา : (http://www.semwebcentral.org/frs/?group_id=89&release_id=380)

```
<z:row NAICSCODE='11' NAICSTEXT='AGRICULTURE, FORESTRY, FISHING AND HUNTING'/>
        <z:row NAICSCODE='111' NAICSTEXT='CROP PRODUCTION'/>
        <z:row NAICSCODE='1111' NAICSTEXT='OILSEED AND GRAIN FARMING'/>
        <z:row NAICSCODE='11111' NAICSTEXT='SOYBEAN FARMING' SICCODE='0116'
SICTEXT='SOYBEANS'/>
        <z:row NAICSCODE='11112' NAICSTEXT='OILSEED (EXCEPT SOYBEAN) FARMING'
SICCODE='0119' SICTEXT='CASH GRAINS, NEC (OILSEED, EXCEPT SOYBEAN FARMING)'/>
        <z:row NAICSCODE='11113' NAICSTEXT='DRY PEA AND BEAN FARMING' SICCODE='0119'
SICTEXT='CASH GRAINS, NEC (DRY PEA AND BEAN FARMS)'/>
        <z:row NAICSCODE='11114' NAICSTEXT='WHEAT FARMING' SICCODE='0111'
SICTEXT='WHEAT'/>
        <z:row NAICSCODE='11115' NAICSTEXT='CORN FARMING' SICCODE='0115'
SICTEXT='CORN'/>
        <z:row NAICSCODE='11115' NAICSTEXT='CORN FARMING' SICCODE='0119' SICTEXT='CASH
GRAINS, NEC (POPCORN FARMING)'/>
        <z:row NAICSCODE='11116' NAICSTEXT='RICE FARMING' SICCODE='0112' SICTEXT='RICE'/>
        <z:row NAICSCODE='11119' NAICSTEXT='OTHER GRAIN FARMING'/>
        <z:row NAICSCODE='111191' NAICSTEXT='OILSEED AND GRAIN COMBINATION FARMING'
SICCODE='0119' SICTEXT='CASH GRAINS, NEC (OILSEED AND GRAIN COMBINATION FARMS)'/>
```

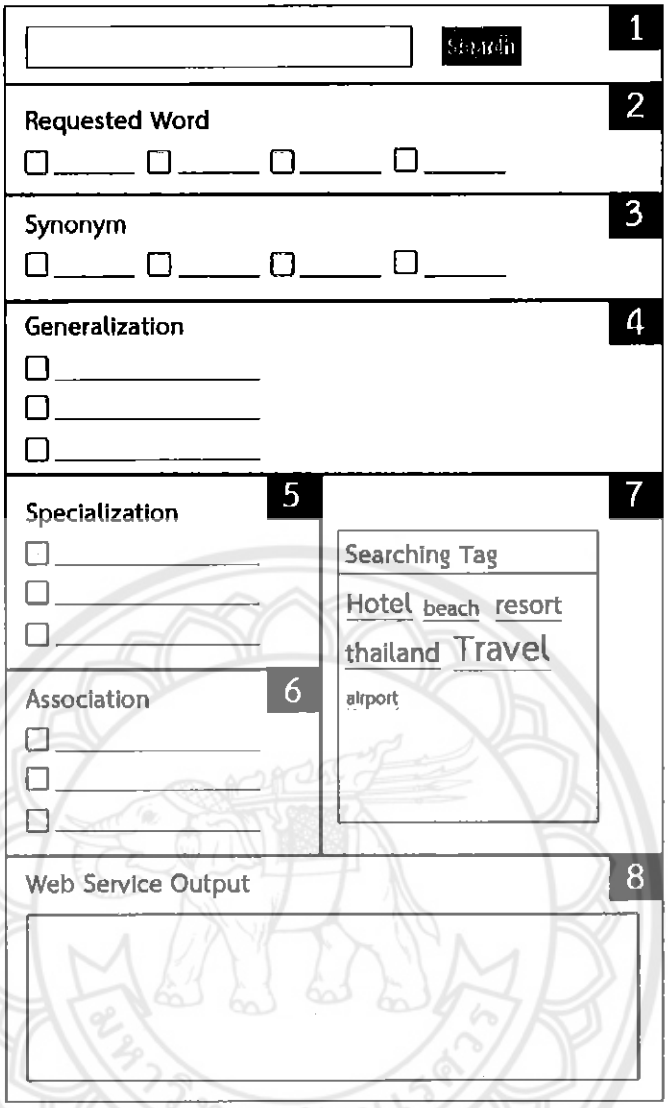
รูปที่ 4.6 แสดงตัวอย่างไฟล์ของ NAICS Ontology



รูปที่ 4.7 แสดงแผนภูมิต้นไม้ของ Travel Ontology

4.4 การออกแบบส่วนแสดงผล

ระบบค้นหาเว็บเซอร์วิสประกอบด้วย 2 ส่วนใหญ่ ๆ คือ ระบบการแท็ก (tag system) และระบบการค้นหาเว็บเซอร์วิส ในที่นี้ผู้จัดทำได้ทำการพัฒนาในส่วนของระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ ซึ่งมีขั้นตอนการทำงานของระบบดังนี้



รูปที่ 4.8 แสดงโครงสร้างจำลองของหน้าเว็บที่ออกแบบขึ้น

จากรูปที่ 4.8 สามารถอธิบายส่วนต่างๆ ของโครงสร้างได้ดังนี้

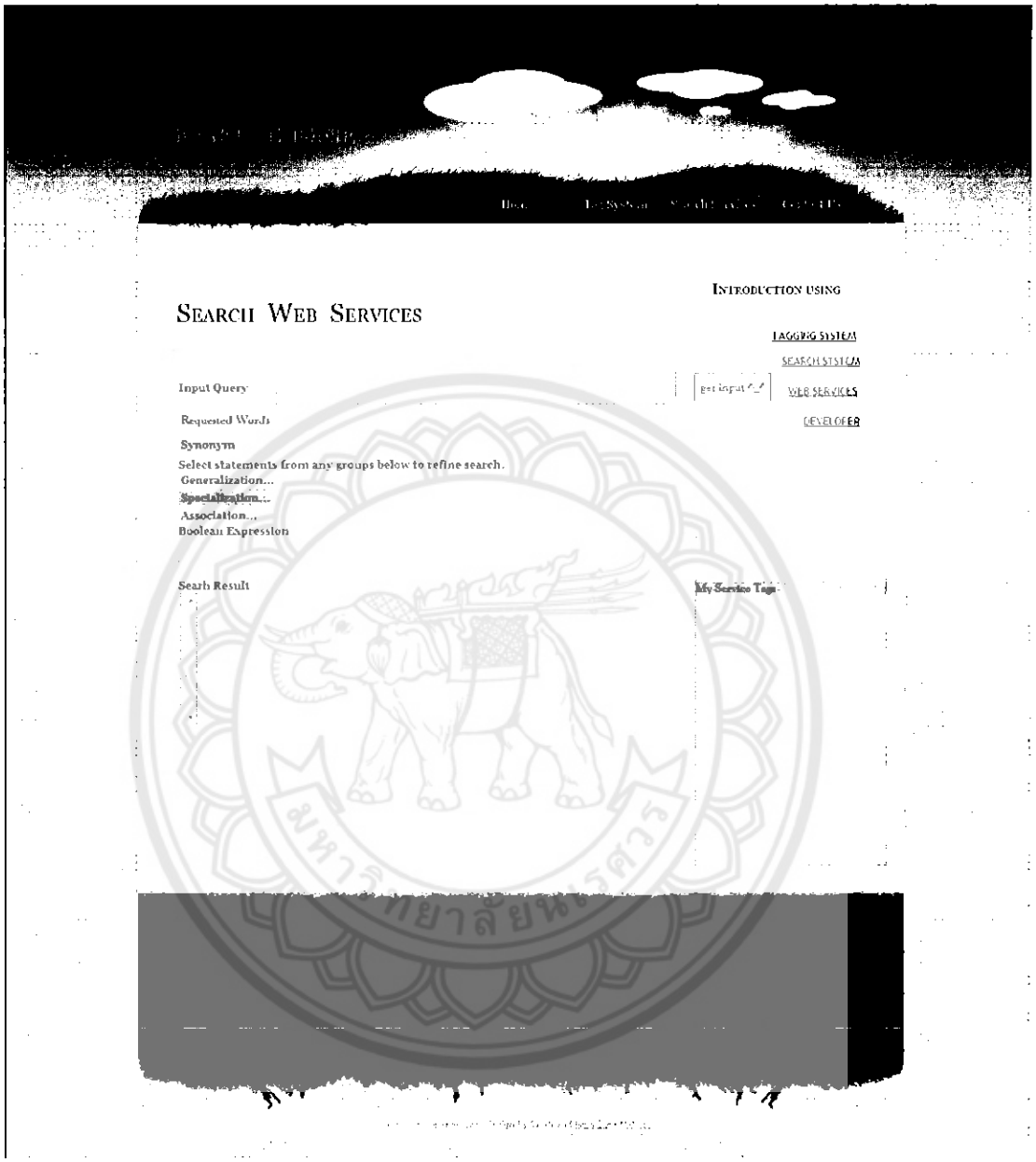
1. ส่วนรับคำหรือประโยคค้นหา
2. ส่วนแสดงคำนามจากประโยคค้นหา
3. ส่วนแสดงคำที่มีความหมายเหมือนกันกับคำที่ถูกเลือก
4. ส่วนส่วนแสดงประโยคแนะนำ โดยข้อมูลจากออนโทโลยี (ontology) แบบ generalization
5. ส่วนแสดงประโยคแนะนำ โดยข้อมูลจากออนโทโลยี (ontology) แบบ specialization
6. ส่วนส่วนแสดงประโยคแนะนำ โดยข้อมูลจากออนโทโลยี (ontology) แบบ Association
7. ส่วนแสดงแท็กของกลุ่มของเว็บเซอร์วิสที่ค้นหาในรูปแบบของ tag cloud
8. ส่วนแสดงผลการค้นหา

4.5 การค้นหาคำที่มีความหมายเหมือนกัน (Synonym)

การค้นหาคำที่มีความหมายเหมือนกัน(Synonym) ได้ประยุกต์ใช้ WordNet API ตัวอย่างของ Synonym จากคำว่า car ต่อไปนี้ (สำหรับตัวอย่างโค้ด อยู่ที่ภาคผนวก ก)

รูปที่ 4.9 แสดงคำที่มีความหมายเหมือนกัน(Synonym) ของคำว่า car จากการประยุกต์ใช้ WordNet API [17]

4.6 ขั้นตอนของการพัฒนาระบบ



รูปที่ 4.10 แสดงหน้าเว็บของระบบค้นหาเว็บเซอร์วิสทั้งหมด

จากรูปข้างต้น แสดงหน้าเว็บของระบบค้นหาเว็บเซอร์วิสทั้งหมดที่ได้ทำการพัฒนาขึ้น แบ่งตามส่วนการทำงานดังต่อไปนี้

4.6.1 ส่วนเลือกคำนามจากประโยคค้นหา

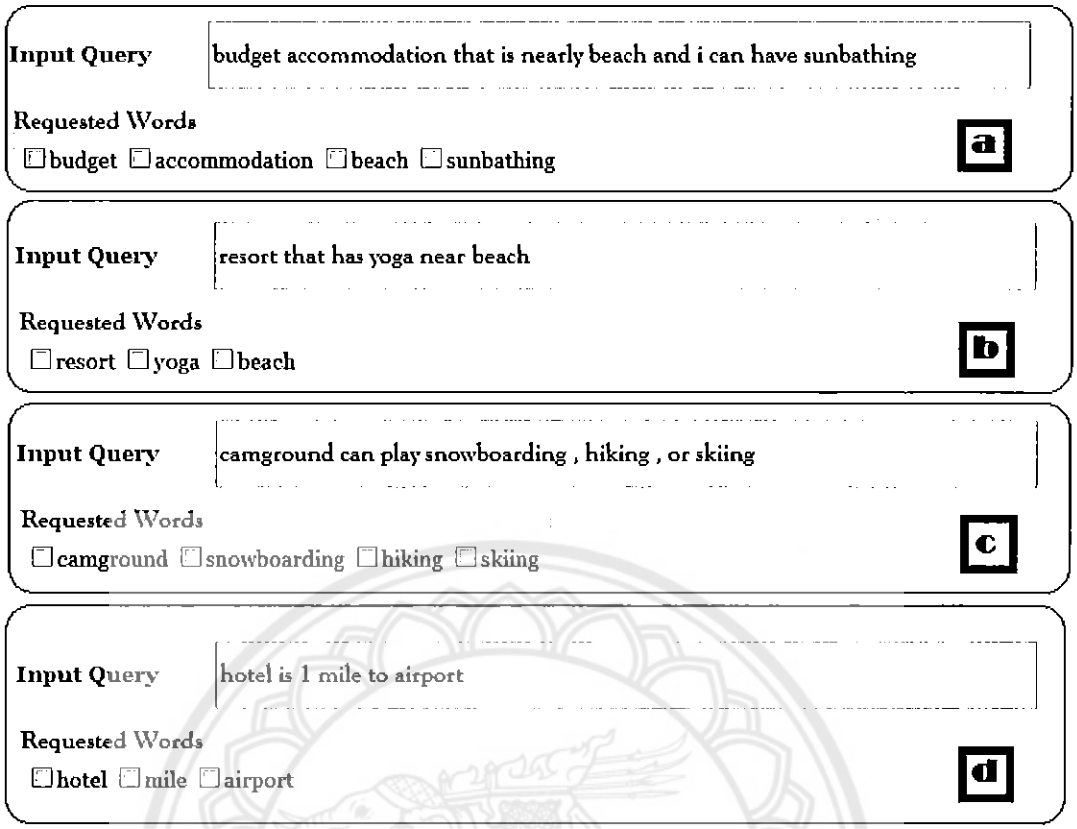
ประยุกต์ใช้ Stanford Log-linear Part-Of-Speech Tagger เพื่อแยกประเภทของคำ และออกแบบโปรแกรมเลือกเฉพาะคำนาม เนื่องจากแอพพลิเคชั่น(Application) ดังกล่าวถูกเขียนด้วยภาษาจาวา (Java) แต่การพัฒนาระบบใช้ภาษา C# ดังนั้นการเชื่อมต่อการทำงานระหว่าง Stanford Log-linear Part-Of-Speech Tagger กับระบบ จึงต้องใช้การ run command line ใน C#

ตัวอย่างโค้ดส่วนเชื่อมต่อการทำงานระหว่าง Stanford Log-linear Part-Of-Speech Tagger กับระบบ โดยการ run command line ใน C#

```
public string connectToPOSTagger(string fileInput)
        {
            Process proc = new Process();
            StreamReader outputStream = StreamReader.Null;
            string output = "";
            try
            {
                //Arguments
                proc.StartInfo.FileName = "java";
                proc.StartInfo.Arguments = " -mx300m  -classpath stanford-
postagger.jar edu.stanford.nlp.tagger.maxent.MaxentTagger -model
models/left3words-wsj-0-18.tagger  -textFile " + fileInput;
                //Option
                proc.StartInfo.CreateNoWindow = true;
                proc.StartInfo.WindowStyle = ProcessWindowStyle.Normal;
                proc.StartInfo.UseShellExecute = false;
                proc.StartInfo.RedirectStandardOutput = true;

                HttpRequest request = HttpContext.Current.Request;

                string appPath = request.MapPath(request.ApplicationPath);

                string directory = System.IO.Path.Combine(appPath,
"App_Data/POStagger");
                DirectoryInfo currentDir = new DirectoryInfo(directory);

                proc.StartInfo.WorkingDirectory = currentDir.FullName;
                proc.Start();

                //Manage output
                outputStream = proc.StandardOutput;
                output = outputStream.ReadToEnd();


            }
            catch (Exception ex)
            {
                if (!proc.Start())
                    throw new Exception("Failed to start Pull process");
            }

            return output;
        }
```

Pseudo code ของโปรแกรมเลือกคำนามจากประโยคค้นหา

```
send data to Stanford  Log-linear Part-Of-Speech Tagger
recieve output (word , partOfSpeech)
      IF output.partOfSpeech = nn or output.partOfSpeech = nnp
         output.partOfSpeech = nns or output.partOfSpeech = nnps
      choose  output.word
      ELSE
      discard output.word
     ENDIF
```



รูปที่ 4.11 แสดง Request word ซึ่งระบบเลือกเฉพาะคำนาม เมื่อประโยคค้นหา ดังต่อไปนี้

a) budget accommodation that is nearly beach and i can have sunbathing

b) resort that has yoga near beach

c) camground can play snowboarding , hiking , or skiing

d) hotel is 1 mile to airport

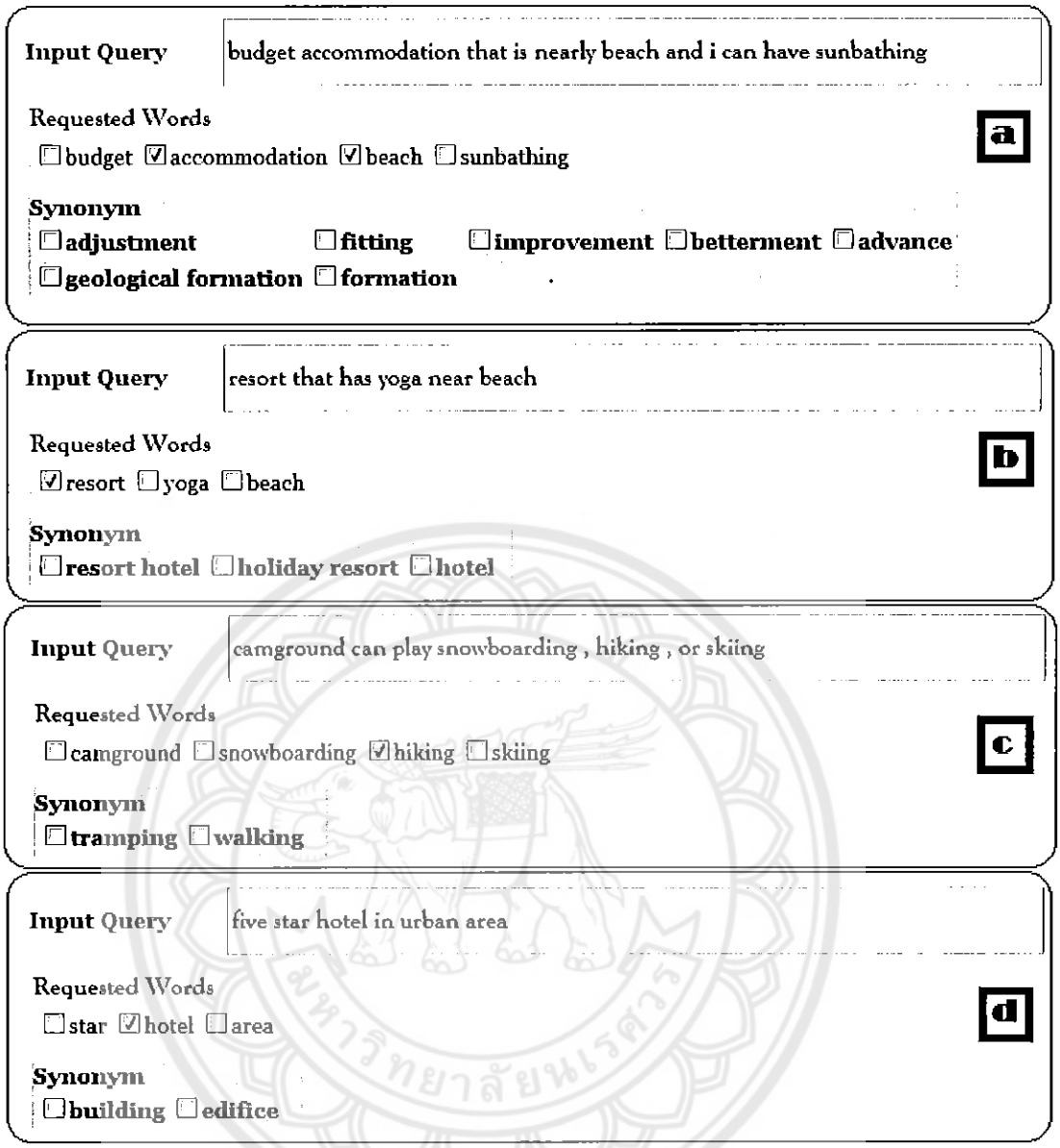
4.6.2 ส่วนนำคำค้นหาที่ผู้ใช้เลือกมาหาคำที่มีความหมายเหมือนกัน

ประยุกต์ใช้ WordNet API เพื่อนำคำค้นหาที่ผู้ใช้เลือกมาหาความหมายนัยต่างๆ และเลือกคำที่มีความหมายเหมือนกับความหมายนัยนั้น จากนั้นนำมาแสดงบนส่วนต่อประสานระบบเพื่อให้ผู้ใช้เลือกต่อไป

ตัวอย่างโค้ดส่วนเชื่อมต่อการทำงานกับ WordNet API เพื่อหาคำที่มีความหมาย

เหมือนกัน

```
public List<string> synonymChoosing(string inputQuery)
{
  List<string> result = new List<string>();
  SpellEngine engine = new SpellEngine();
  LanguageConfig enConfig = new LanguageConfig();

  HttpRequest request = HttpContext.Current.Request;
  string appPath = request.MapPath(request.ApplicationPath);

  enConfig.LanguageCode = "en";
  enConfig.HunspellAffFile = System.IO.Path.Combine(appPath,
                              "App_Data/wordNetAPI/en_us.aff");
  enConfig.HunspellDictFile =System.IO.Path.Combine(appPath,
                              "App_Data/wordNetAPI/en_us.dic");
  enConfig.HunspellKey = "";
  enConfig.HyphenDictFile = System.IO.Path.Combine(appPath,
                             "App_Data/wordNetAPI/hyph_en_us.dic");
  enConfig.MyThesIdxFile = System.IO.Path.Combine(appPath,
                            "App_Data/wordNetAPI/th_en_us_new.idx");
  enConfig.MyThesDatFile = System.IO.Path.Combine(appPath,
                            "App_Data/wordNetAPI/th_en_us_new.dat");
  engine.AddLanguage(enConfig);
  try
  {
  ThesResult tr = engine["en"].LookupSynonyms(inputQuery, true);
  if (tr.IsGenerated)
       Console.WriteLine("Generated over stem (The original word form wasn't
in the thesaurus)");
       int i = 1;
       foreach (ThesMeaning meaning in tr.Meanings)
       {
           if (i == 1)
           {
              foreach (string synonym in meaning.Synonyms)
              {
                    result.Add(synonym);
              }
              i = -1;
           }
       }
  }
  catch (Exception ex)
  {
  }
  foreach (string syn in result)
  {
       Console.Write(syn+"  ");
  }
  return result;
  }
}
```



รูปที่ 4.12 แสดง Synonym ของคำต่อไปนี้ที่ผู้ใช้เลือก ดังต่อไปนี้

a) accommodation และ beach b) resort c) hiking d) hotel

4.6.3 ส่วนปรับปรุงประโยคโดยข้อมูลจากออนโทโลยี (ontology)

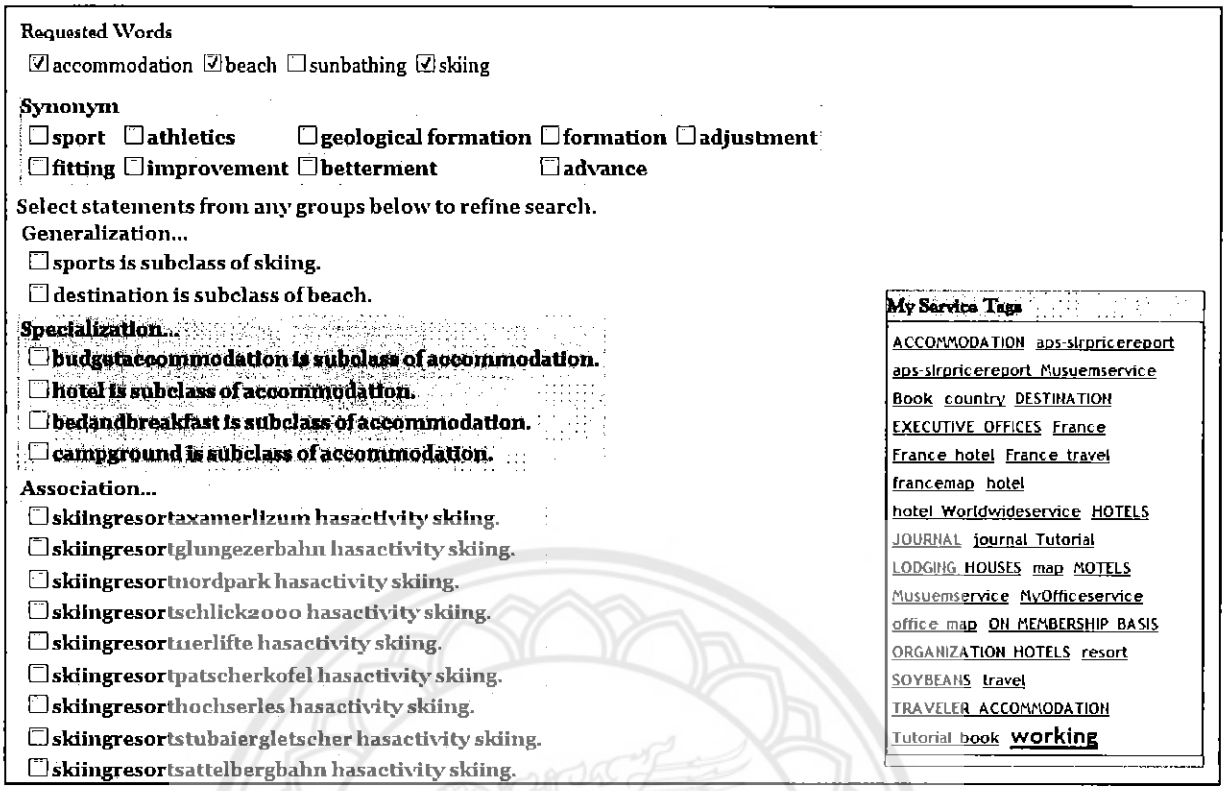
เนื่องจากออนโทโลยี มีโครงสร้างข้อมูลแบบลำดับชั้น ทำให้เราสามารถสร้างแผนภูมิต้นไม้ (Tree) ของความสัมพันธ์ภายในออนโทโลยีได้ และนำแผนภูมิต้นไม้ดังกล่าวไปสร้างเป็นประโยคแนะนำให้แก่ผู้ใช้ ทำให้ประโยคที่ได้มีความสอดคล้องกับออนโทโลยี เมื่อผู้ใช้เลือกประโยคแนะนำจะทำให้การค้นหาตรงกับข้อมูลที่ระบบมีอยู่มากขึ้น และประสิทธิภาพในการค้นหาจะเพิ่มมากยิ่งขึ้น

Pseudo code ของโปรแกรมสร้างแผนภูมิต้นไม้โดยข้อมูลจากออนโทโลยี (ontology)

```
parse OWL()
       IF node don't exist
          Add new node (node name, relation, next node)
       ELSE
          Find node
          Add next node and relation
      ENDIF
```

Pseudo code ของสร้างประโยคโดยแผนภูมิต้นไม้ของออนโทโลยี (ontology)

```
get input word
       FOR all node in tree
          IF inputword=nodeName
             getSpecialization(nodeName)
            getGeneralization(nodeName)
             getAssociation(nodeName)
          ENDIF
       END FOR
```



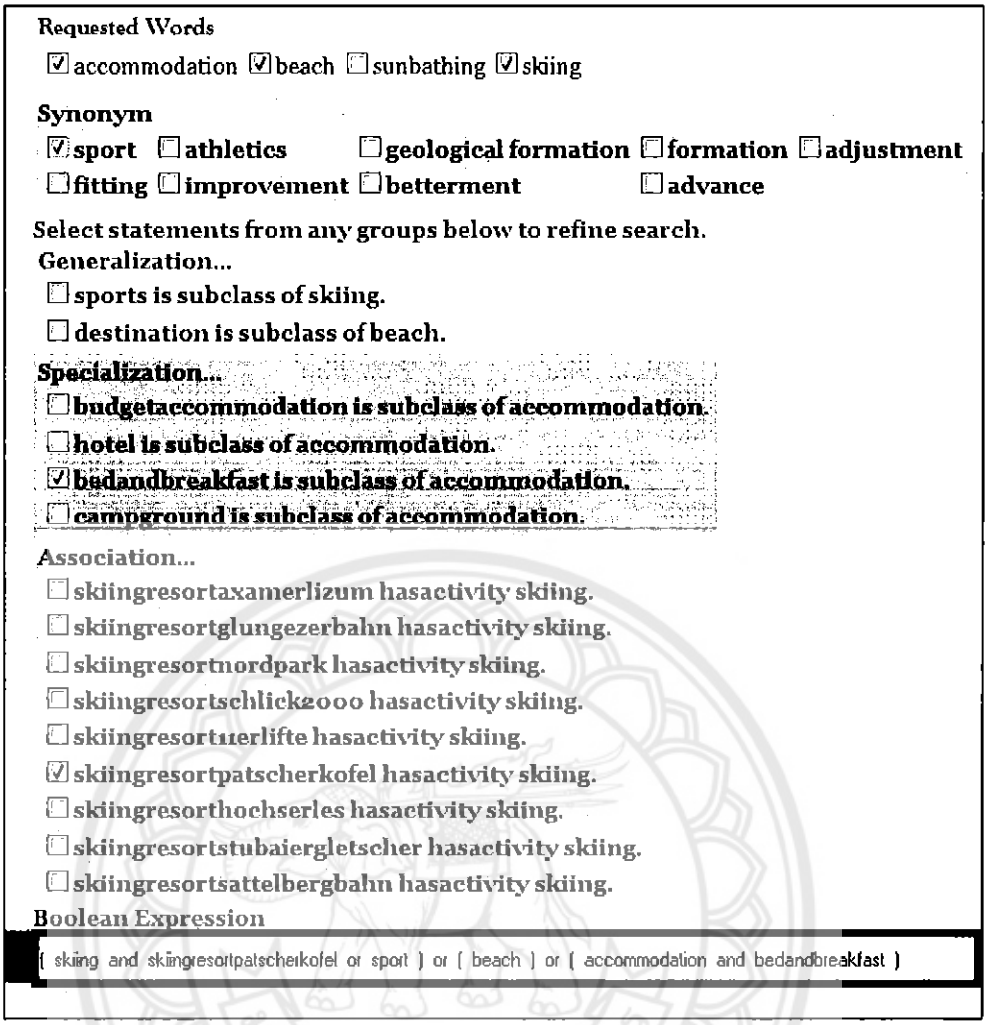
รูปที่ 4.13 แสดงประโยคแนะนำของการเลือก Request word : accommodation , beach และ skiing

4.6.4 ส่วนสร้างและแสดงนิพจน์ทางตรรกศาสตร์(Boolean Expression)

เมื่อผู้ใช้เลือกข้อมูลต่างๆ จะมีการสร้างนิพจน์ทางตรรกศาสตร์(Boolean Expression) ตามเงื่อนไขที่ถูกกำหนดไว้ และนำนิพจน์ทางตรรกศาสตร์(Boolean Expression)ไปใช้เป็นเงื่อนไขในการเลือกเว็บเซอร์วิสซึ่งเป็นผลของการค้นหา

Pseudo code ของโปรแกรมสร้างและแสดงนิพจน์ทางตรรกศาสตร์(Boolean Expression)

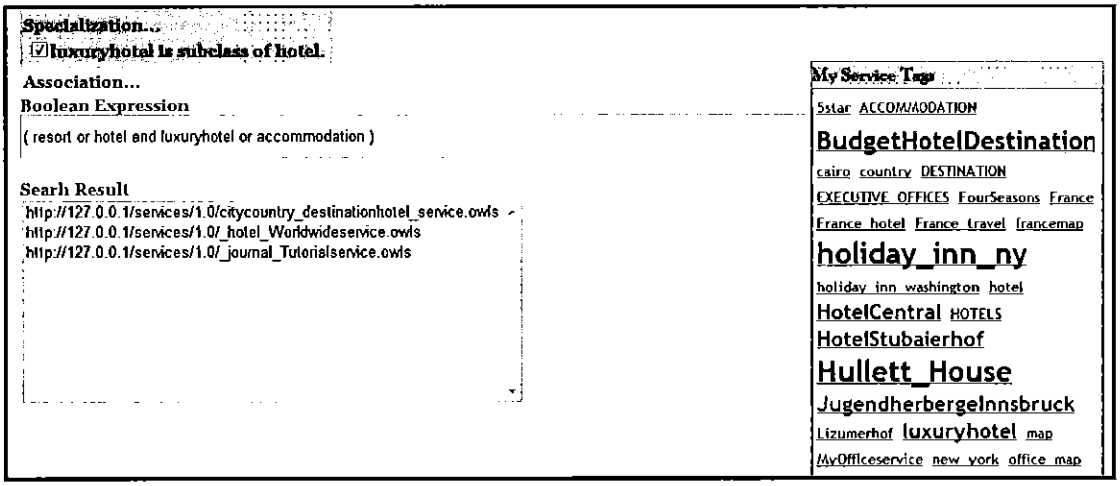
```
get input from user
       FOR all word that user choose
       IF word is keyword
             buffer = "( "
       IF word is Requested Word
             buffer = node;

       ELSE IF word is Synonym
             buffer = " or " + node;

       ELSE IF word is Specialization
             buffer = " or " + node;

       ELSE IF word is Generalization
             buffer = " and " + node;

       ELSE IF word is Association
             buffer = " and " + node;

     END IF

ADD buffer into List Boolean Expresion
ADD  ")"  into List Boolean Expresion

       END FOR
checkcloseParentheses = false;
checkopenParentheses = false;

       FOR all result (List Boolean Expresion) , i < result.Count
             IF data="( "
                   checkopenParentheses = true
             END IF

             IF data=") "
                   checkcloseParentheses = true
             END IF

             IF checkopenParentheses= true
                   buffBE = buffBE + result[i]
             END IF

             IF( checkopenParentheses= true
                &&checkcloseParentheses= true
                && i!=result.Count-1)

                         buffBE = buffBE + " or "
                         checkcloseParentheses = false
                         checkopenParentheses = false
             END IF

             IF( checkopenParentheses= true
                &&checkcloseParentheses= true
                && i=result.Count-1)

                         buffBE = buffBE
                         checkcloseParentheses = false
                         checkopenParentheses = false
             END IF

             END FOR
return buffBE
```



รูปที่ 4.14 แสดงนิพจน์ทางตรรกศาสตร์ของการเลือก Request word : hotel Synonym : sports Specialization : bedandbreakfast และ association : skiingresortpatscherkofel

4.6.5 แสดงแท็กของกลุ่มของเว็บเซอร์วิสที่ค้นหาในรูปแบบของ tag cloud

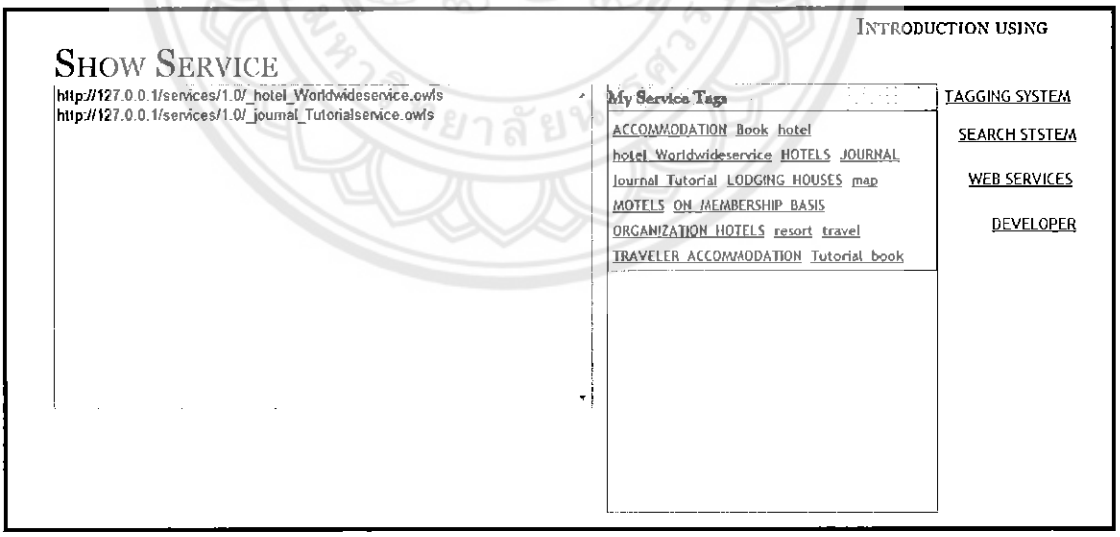
นอกจากการค้นหาโดยใช้คำหรือประโยคค้นหา (Keyword Search) แล้วระบบยังสร้างส่วนเสริมให้แก่ผู้ใช้ โดยนำกลุ่มของเว็บเซอร์วิสที่ได้จากการค้นหาแต่ละครั้งมาหาแท็กแล้วแสดงรูปแบบของ tag cloud แท็กเหล่านี้ได้มาจากการแท็กโดยผู้ใช้ โดยคำแต่ละคำจะมีความถี่ในการแท็กที่แตกต่างกัน ใน tag cloud แท็กที่มีความถี่มากจะมีขนาดตัวอักษรที่ใหญ่กว่าแท็กที่มีความถี่น้อย รวมถึงความเข้มและความหนาของตัวอักษรด้วย



รูปที่ 4.15 แสดงแท็กของกลุ่มของเว็บเซอร์วิสที่ค้นหาในรูปแบบของ tag cloud

4.6.6 ส่วนการตอบสนองต่อการเลือกแท็กใน tag cloud

เมื่อผู้ใช้คลิ้กเลือกแท็กใน tag cloud ระบบจะแสดงกลุ่มของเว็บเซอร์วิสที่ถูกแท็กด้วยคำนั้นทั้งหมด เนื่องจากการทำงานของระบบเป็นแบบอินเตอร์แอคทีฟ หลังจากผู้ใช้เลือกแท็กใดๆ แล้ว tag cloud ก็จะเปลี่ยนแปลงตามไปด้วย โดยแท็กใน tag cloud ที่ปรับปรุงใหม่จะได้มาจากกลุ่มของเว็บเซอร์วิสที่ถูกแท็กด้วยคำที่ถูกเลือกก่อนหน้านั้น



รูปที่ 4.16 แสดงรายละเอียดของแท็กที่ผู้ใช้คลิก

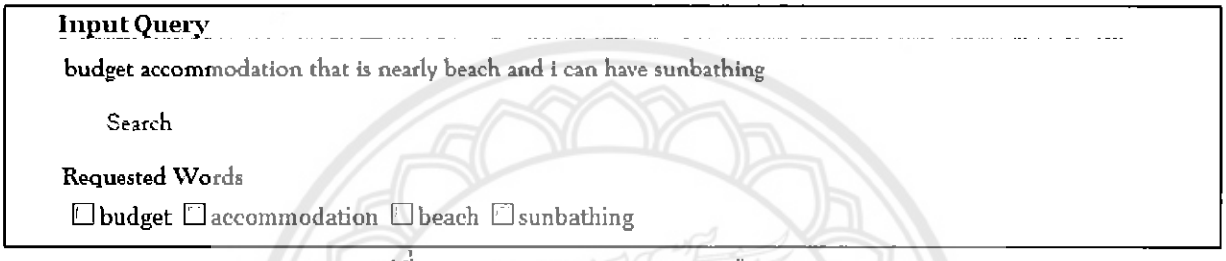
4.7 ขั้นตอนการทำงานขอบระบบ

จากการทดสอบพบว่าระบบมีความเปลี่ยนแปลงทุกครั้งที่ผู้ใช้ตอบสนองต่อระบบ

โดยระบบมีขั้นตอนการทำงานและมีการเปลี่ยนแปลง เมื่อผู้ใช้ตอบสนองต่อระบบ ดังต่อไปนี้

4.7.1 กรอกคำหรือประโยคค้นหา และกดปุ่มค้นหา

a) ระบบจะแยกประเภทของคำและแสดงเฉพาะคำนามซึ่งถือเป็นคำที่ผู้ใช้ร้องขอ (request word) และแสดงผลบนหน้าเว็บในส่วนของ Request word



รูปที่ 4.17 แสดงผลหลังจากการเลือก Request word

4.7.2 เลือกคำนามจากคำหรือประโยคค้นหา

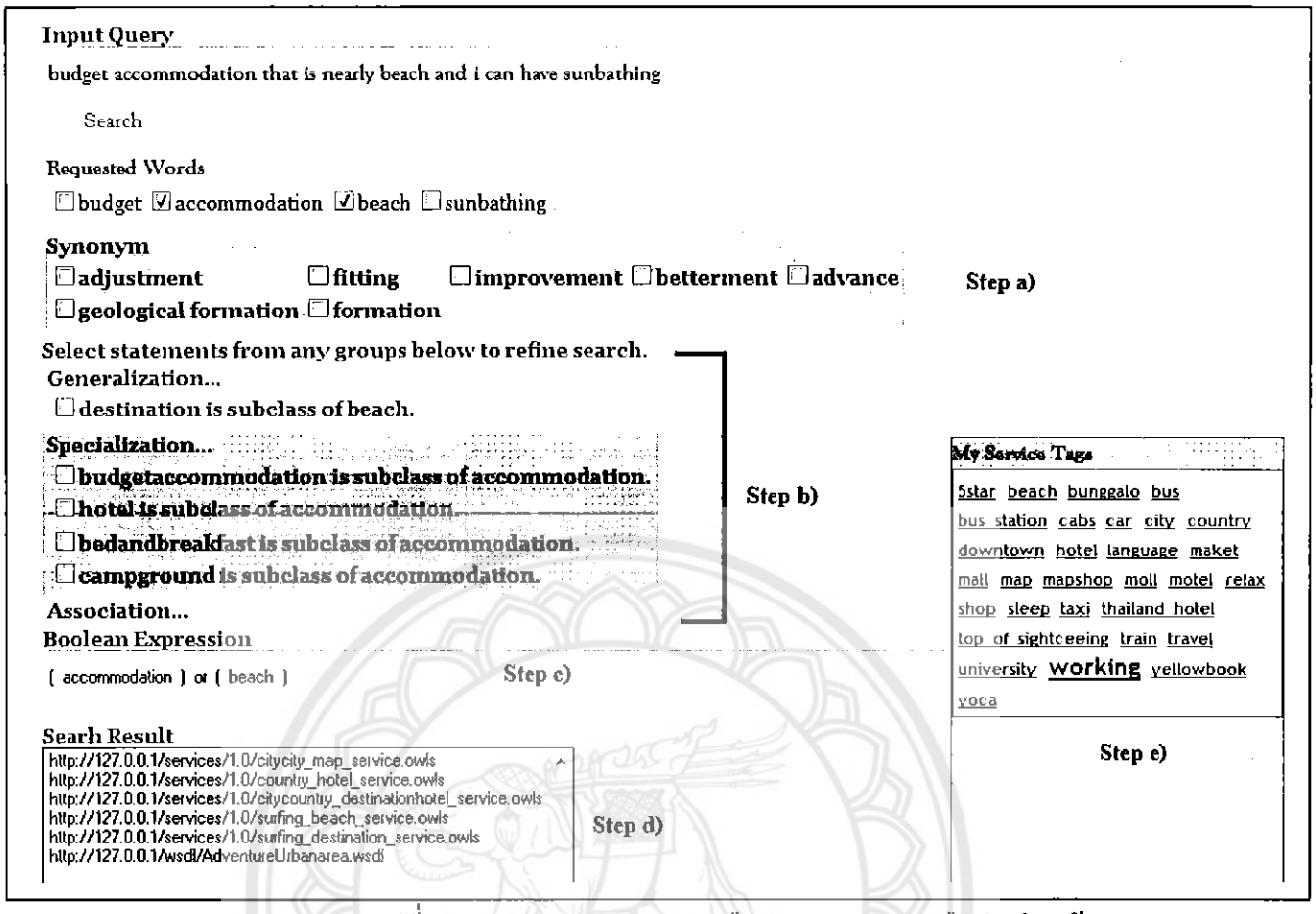
a) ระบบจะค้นหาคำที่มีความหมายเหมือนกันกับคำที่ถูกเลือก และแสดงผลบนหน้าเว็บในส่วนของ Synonym

b) ระบบสร้างประโยคแนะนำจากออนโทโลยี และแสดงผลบนหน้าเว็บในส่วนของ Generalization Specialization และ Association

c) ระบบจะสร้างและแสดงนิพจน์ทางตรรกศาสตร์ (Boolean Expression)

d) ระบบแสดงผลของการค้นหา

e) ระบบแสดง tag cloud ของกลุ่มเว็บเซอร์วิสของผลการค้นหา



รูปที่ 4.18 แสดงผลหลังจากการเลือกคำนามจากคำหรือประโยคค้นหา

4.7.3 เลือกคำที่มีความหมายเหมือนกัน (Synonym) ของคำที่ถูกเลือก

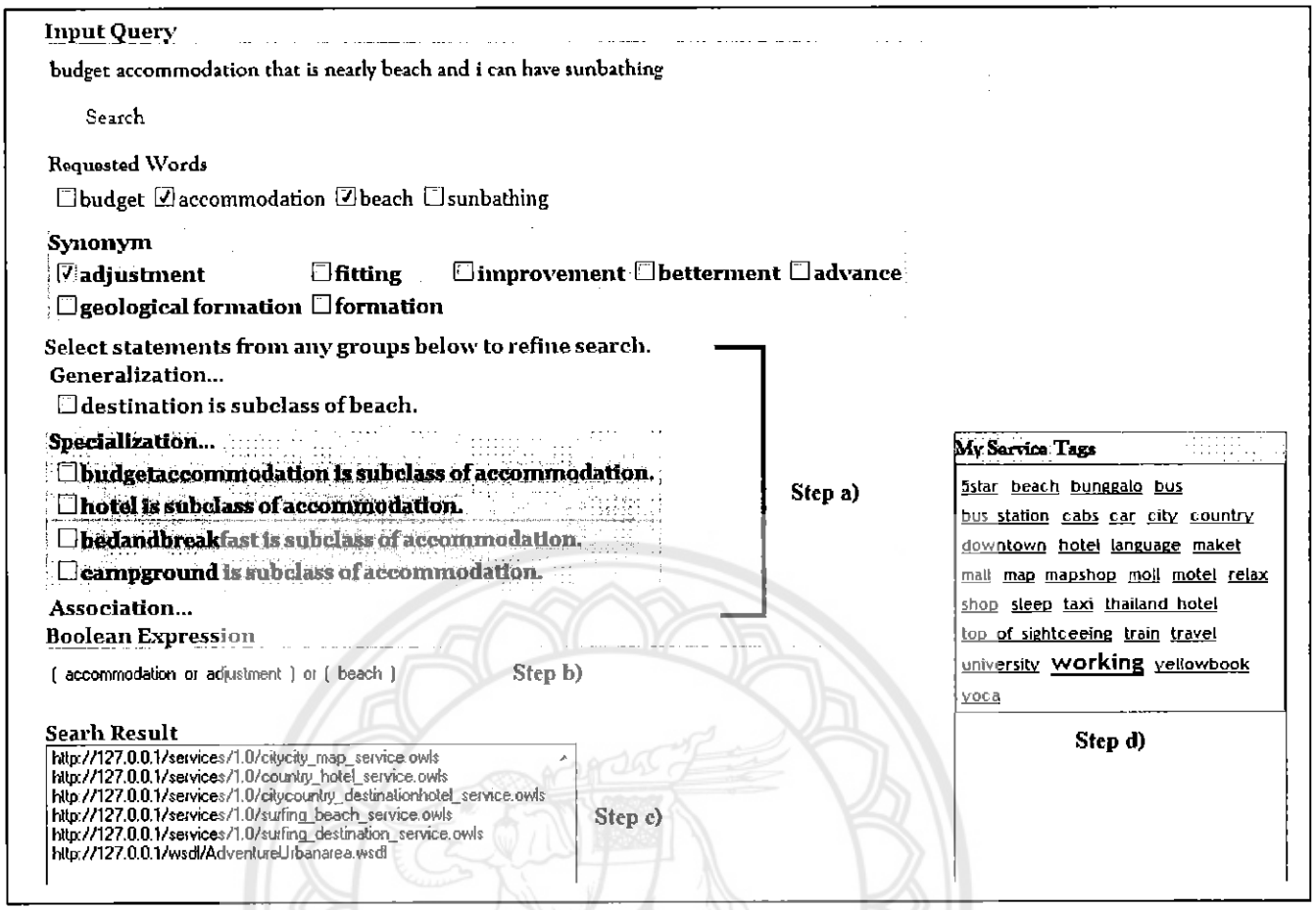
a) ระบบสร้างประโยคแนะนำจากออนโทโลยี และแสดงผลบนหน้าเว็บในส่วน

ของ Generalization Specialization และ Association

b) ระบบจะสร้างและแสดงนิพจน์ทางตรรกศาสตร์ (Boolean Expression)

c) ระบบแสดงผลของการค้นหา

d) ระบบแสดง tag cloud ของกลุ่มเว็บเซอร์วิสของผลการค้นหา



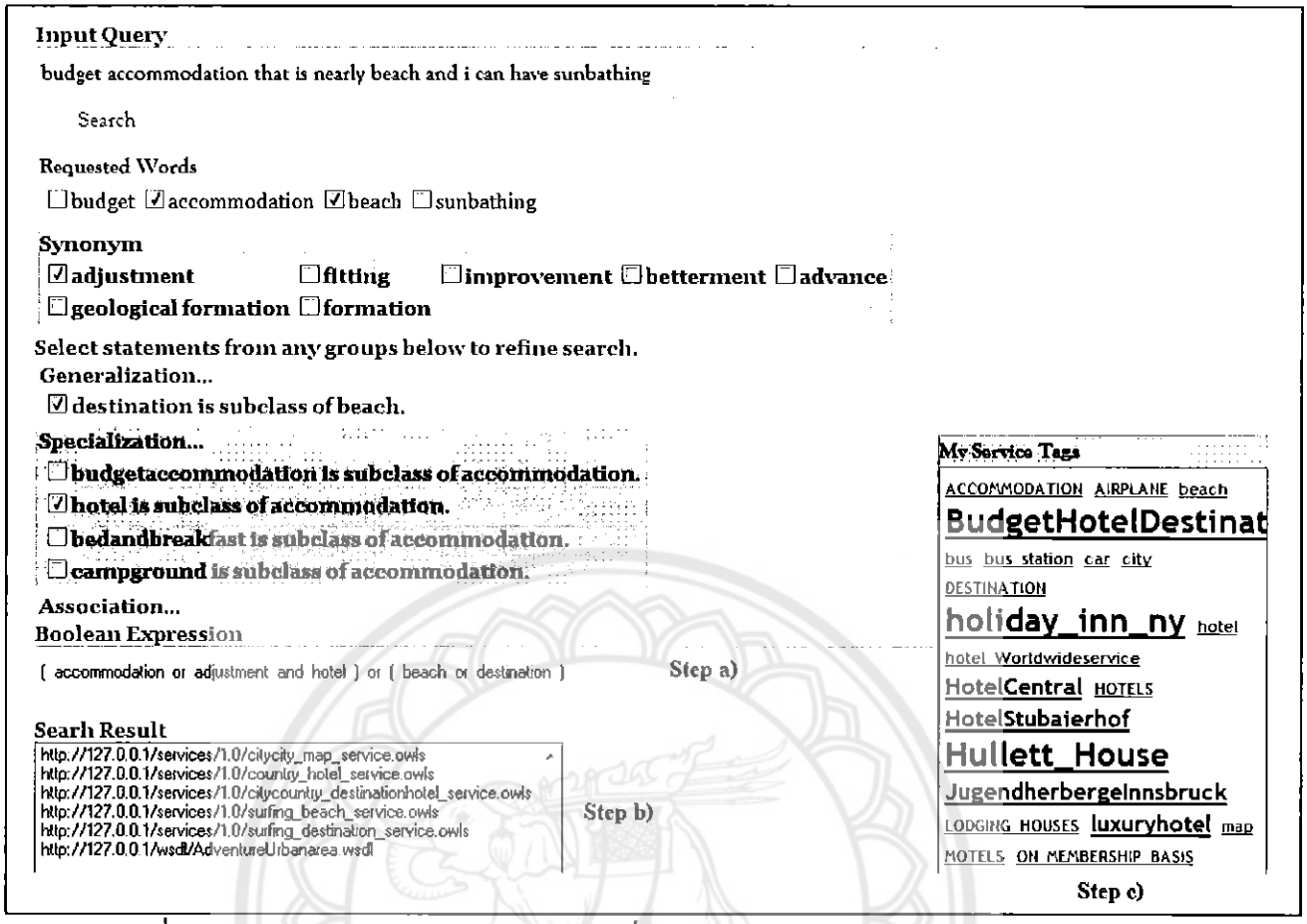
รูปที่ 4.19 แสดงผลหลังจากการเลือกคำที่มีความหมายเหมือนกัน (Synonym)

4.7.4 เลือกประโยคที่ถูกปรับปรุงโดยข้อมูลจากออนโทโลยี (ontology)

a) ระบบจะสร้างและแสดงนิพจน์ทางตรรกศาสตร์ (Boolean Expression)

b) ระบบแสดงผลของการค้นหา

c) ระบบแสดง tag cloud ของกลุ่มเว็บเซอร์วิสของผลการค้นหา

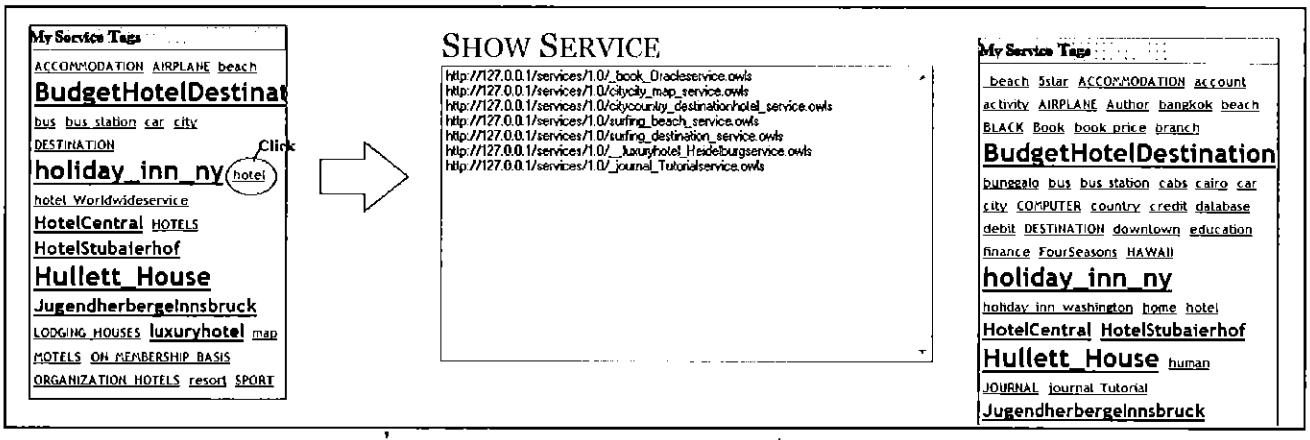


รูปที่ 4.20 แสดงผลหลังจากการเลือกประโยคที่ถูกปรับปรุงโดยข้อมูลจากออนโทโลยี (ontology)

4.7.5 เลือกแท็กใน tag cloud

- ระบบจะแสดงกลุ่มของเว็บเซอร์วิสที่ถูกแท็กด้วยคำนั้นทั้งหมด

a) ระบบแสดง tag cloud ที่ปรับปรุงใหม่ โดยแท็กใน tag cloud ที่ปรับปรุงใหม่จะได้แท็กมาจากกลุ่มของเว็บเซอร์วิสที่ถูกแท็กด้วยคำที่ถูกเลือกก่อนหน้านั้น



รูปที่ 4.21 แสดงผลหลังจากการเลือกแท็กใน tag cloud

รูปที่ 4.22 แสดงสรุปขั้นตอนการทำงานของระบบ

บทที่ 5

ผลการทดลอง

เนื่องจากระบบค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ เป็นระบบที่ตอบสนองต่อการใช้งานของผู้ใช้ในการค้นหา รวมทั้งให้คำแนะนำในการค้นหาแก่ผู้ใช้ โดยทำงานแบบอินเตอร์แอคทีฟ การทดสอบระบบจึงต้องครอบคลุมทั้งในด้านการทำงาน การแสดงผล และความเร็วในการประมวลผล จึงได้มีการสร้างแบบสอบถามเพื่อประเมินผลความพึงพอใจของผู้ใช้เมื่อได้ทดลองใช้งานระบบ และการทดสอบเวลาการทำงานของระบบต่าง ๆ

5.1 ทดสอบความพึงพอใจของผู้ใช้ต่อระบบ

ตัวอย่างแบบสอบถามระดับความพึงพอใจของผู้ใช้
ระบบค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ

คำชี้แจง

1. ซึ่งแบบสอบถามแบ่งออกเป็น 2 ด้าน คือ
   1.1 ด้านการทำงานได้ตามฟังก์ชันงานของระบบ (Functional Test)
   1.2 ด้านความง่ายต่อการใช้งานระบบ (Usability Test)
2. ในการตอบแบบสอบถามให้ท่านดำเนินการดังนี้
   ทำเครื่องหมาย ✓ ลงในช่องในแบบสอบถามที่ตรงกับระดับความคิดเห็นของท่านมากที่สุด โดยตัวเลขของระดับความพึงพอใจแต่ละด้านมีความหมายดังนี้
   - 5 หมายถึง ความเหมาะสม/ความพึงพอใจในระดับมากที่สุด
   - 4 หมายถึง ความเหมาะสม/ความพึงพอใจในระดับมาก
   - 3 หมายถึง ความเหมาะสม/ความพึงพอใจในระดับปานกลาง
   - 2 หมายถึง ความเหมาะสม/ความพึงพอใจในระดับน้อย
   - 1 หมายถึง ความเหมาะสม/ความพึงพอใจในระดับน้อยที่สุด

เพศ ○ ชาย ○ หญิง

แบบสอบถามความพึงพอใจด้านการทำงานได้ตามฟังก์ชันงานของระบบ (Function Test)

รายการประเมิน	ระดับความพึงพอใจ				
	5	4	3	2	1
1.ความถูกต้องในการแสดงข้อมูล 2.ความถูกต้องของการผลลัพธ์ในการค้นหาเว็บเซอร์วิส 3.ความรวดเร็วในการประมวลผลของระบบ 4.ความน่าเชื่อถือได้ของระบบ 5.การป้องกันข้อผิดพลาดที่อาจเกิดขึ้น					

แบบสอบถามความพึงพอใจด้านความง่ายต่อการใช้งานระบบ (Usability Test)

รายการประเมิน	ระดับความพึงพอใจ				
	5	4	3	2	1
1.ความง่ายต่อการใช้งานของระบบ 2.ความเหมาะสมในการเลือกใช้ชนิดและขนาดของตัวอักษรในการแสดงผล 3.ความเหมาะสมในแสดงผล tag cloud 4.ความเหมาะสมในการใช้ข้อความในอธิบายส่วนต่างๆของโปรแกรมง่ายต่อการเข้าใจ 5.ความเหมาะสมในการปฏิสัมพันธ์โต้ตอบกับผู้ใช้ 6.ความเหมาะสมในการวางตำแหน่งของส่วนประกอบต่างๆในการแสดงผล					

ข้อเสนอแนะอื่นๆ

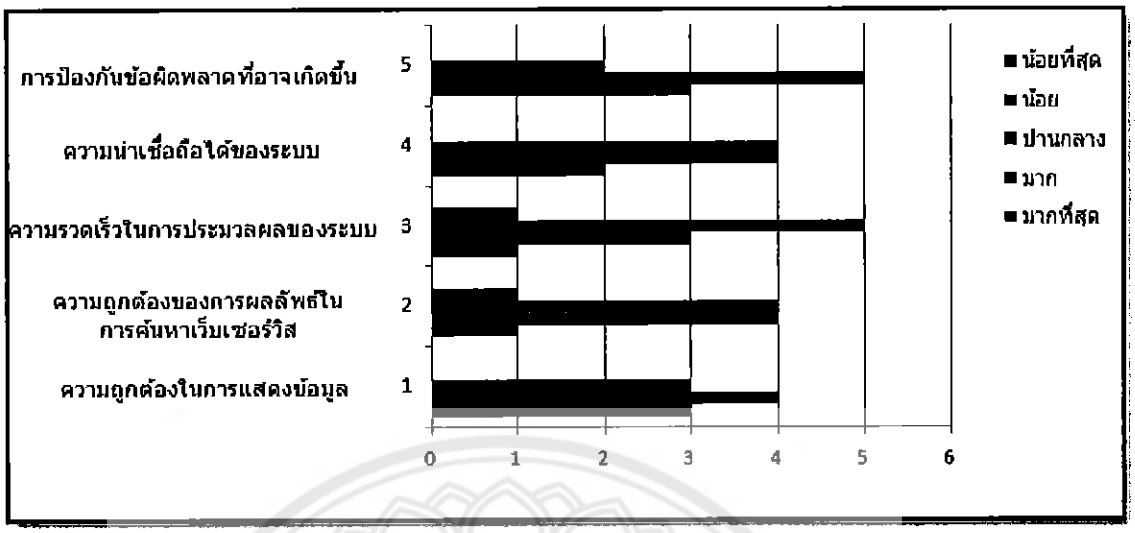
.....

.....

ขอขอบพระคุณเป็นอย่างสูงที่สละเวลาในการให้ข้อมูลที่เป็นจริง

จากการสำรวจระดับความพึงพอใจของผู้ใช้ จำนวน 10 คน เป็นเพศชายจำนวน 4 คน และเพศหญิงจำนวน 6 คน ผลการประเมินแบ่งเป็น 2 ส่วนดังต่อไปนี้

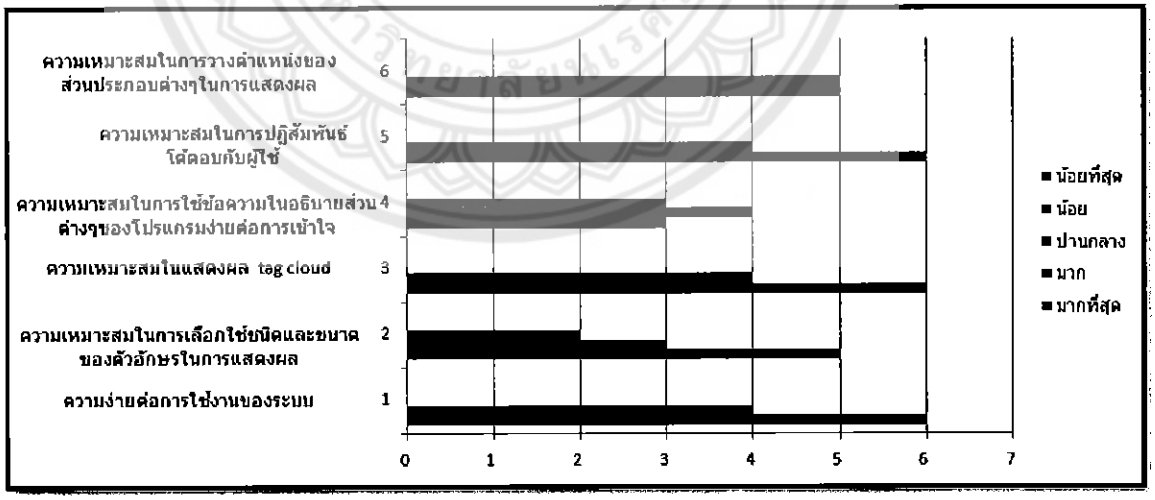
1) ผลการทดสอบความพึงพอใจด้านการทำงานได้ตามฟังก์ชันงานของระบบ (Function Test)



รูปที่ 5.1 แสดงกราฟสรุปผลการสำรวจระดับความพึงพอใจของผู้ใช้ระบบค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ ด้านการทำงานตามฟังก์ชันงานของระบบ(Function Test)

จากรูปที่ 5.1 พบว่า ในส่วนของฟังก์ชันการทำงานนั้น มีผู้ใช้งานพอใจอยู่ในเกณฑ์ที่ดีถึงปานกลาง ในส่วนของความเร็วในการทำงานค่อนข้างจะอยู่ในเกณฑ์ระดับปานกลาง

2) ผลการทดสอบความพึงพอใจด้านความง่ายต่อการใช้งานระบบ (Usability Test)



รูปที่ 5.2 แสดงกราฟสรุปผลการสำรวจระดับความพึงพอใจของผู้ใช้ระบบค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟด้านความง่ายต่อการใช้งานระบบ (Usability Test)

จากรูปที่ 5.2 พบว่า ในส่วนของความง่ายต่อการใช้งานระบบนั้น มีผู้ใช้งานพอใจอยู่ในเกณฑ์ที่ดีถึงดีมาก

5.2 ทดสอบเวลาการทำงานของระบบ

5.2.1 ทดสอบเวลาการทำงานของขั้นตอนการ Get requested words

ตารางที่ 5.1 แสดงเวลาการทำงานของขั้นตอนการ Get request words

ครั้งที่	1	2	3	4	5	6	7	8	9	10
เวลา(วินาที)	5	4	3	4	3	3	4	4	5	4
เวลาเฉลี่ย(วินาที)	3.9									

จากตารางที่ 5.1 การเชื่อมโยงการทำงานของ Stanford Log-linear Part-Of-Speech Tagger และระบบที่พัฒนาขึ้น เพื่อเลือกคำนามจากประโยคค้นหาแต่ละครั้งใช้เวลา 3.9 วินาทีโดยประมาณ จากการทดสอบ 10 ครั้ง

ซึ่งเวลาการทำงานของระบบที่พัฒนาขึ้นจริงๆ โดยไม่ได้รวมเวลาของ Stanford Log-linear Part-Of-Speech Tagger เป็นไปดังตารางต่อไปนี้

ตารางที่ 5.2 แสดงเวลาการทำงานของระบบที่พัฒนาขึ้นในขั้นตอนการ Get request words

ครั้งที่	1	2	3	4	5	6	7	8	9	10
เวลา(วินาที)	0.9	1	1	1	0.9	0.9	1	1	1	1
เวลาเฉลี่ย(วินาที)	0.97									

จากตารางที่ 5.2 จะเห็นว่า เวลาการทำงานของระบบที่พัฒนาขึ้นใช้เวลาเพียง 0.97 วินาที โดยประมาณ จากการทดสอบ 10 ครั้ง

5.2.2 ทดสอบเวลาการทำงานของขั้นตอนการ Get synonyms

ตารางที่ 5.3 แสดงเวลาการทำงานของขั้นตอนการ Get synonyms

ครั้งที่	1	2	3	4	5	6	7	8	9	10
เวลา(วินาที)	9	8	7	7	7	7	8	7	8	8
เวลาเฉลี่ย(วินาที)	7.6									

จากตารางที่ 5.2 การเชื่อมโยงการทำงานของ WordNet API และระบบที่พัฒนาขึ้น เพื่อค้นหาคำที่มีความหมายเหมือนกับคำที่ถูกเลือก แต่ละครั้งใช้เวลา 7.6 วินาทีโดยประมาณ จากการทดสอบ 10 ครั้ง

ซึ่งเวลาการทำงานของระบบที่พัฒนาขึ้นจริงๆ โดยไม่ได้รวมเวลาของ WordNet API เป็นไปดังตารางต่อไปนี้

ตารางที่ 5.4 แสดงเวลาการทำงานของระบบที่พัฒนาขึ้นในขั้นตอนการ Get synonyms

ครั้งที่	1	2	3	4	5	6	7	8	9	10
เวลา(วินาที)	1	1.2	1	1.5	1.5	1.5	1.3	1.2	1	1
เวลาเฉลี่ย(วินาที)	1.22									

จากตารางที่ 5.4 จะเห็นว่า เวลาการทำงานของระบบที่พัฒนาขึ้นใช้เวลาเพียง 1.22 วินาที โดยประมาณ จากการทดสอบ 10 ครั้ง

5.2.3 ทดสอบเวลาการทำงานของขั้นตอนการ Get relationships

ตารางที่ 5.5 แสดงเวลาการทำงานของขั้นตอนการ Get relationships

ครั้งที่	1	2	3	4	5	6	7	8	9	10
เวลา(วินาที)	1	1	1	1	1	1	1	1	1	1
เวลาเฉลี่ย(วินาที)	1									

จากตารางที่ 5.5 การสร้างประโยคแนะนำของคำที่ถูกเลือกจากแผนภูมิต้นไม้ (tree) ของออนโทโลยี (Ontology) แต่ละครั้งใช้เวลา 1 วินาทีโดยประมาณ จากการทดสอบ 10 ครั้ง

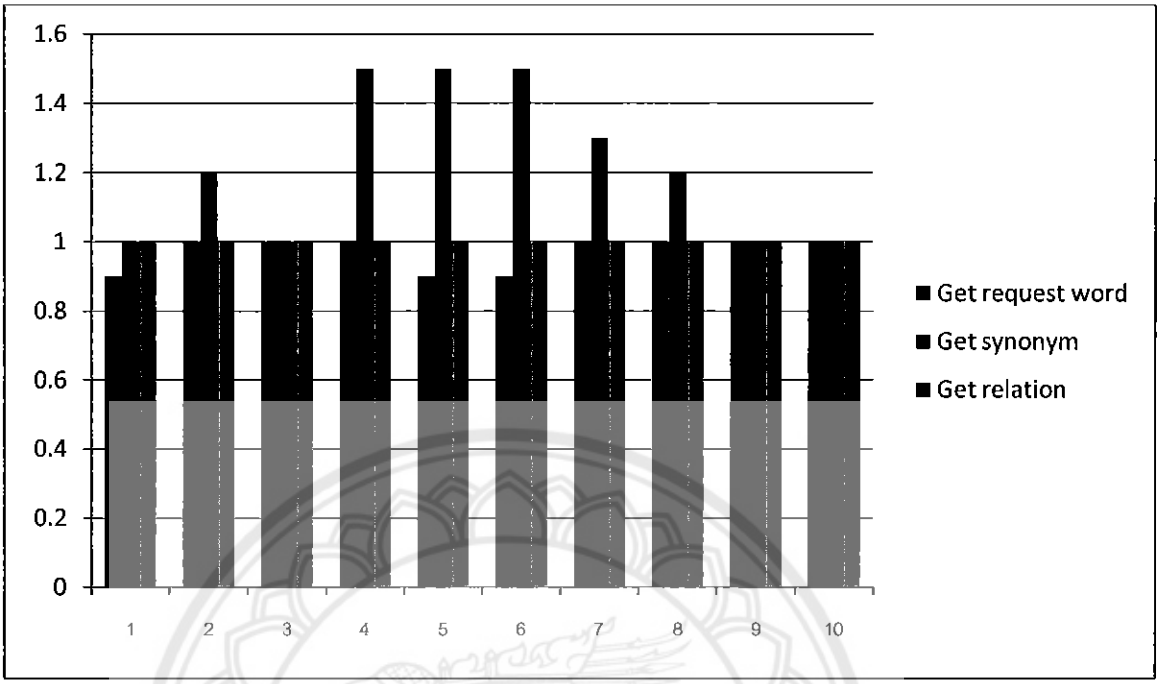
5.2.4 ทดสอบเวลาการทำงานของขั้นตอนการแสดงผลต่างๆ

ตารางที่ 5.6 แสดงเวลาในการแสดงผลลัพธ์ในแบบต่างๆ

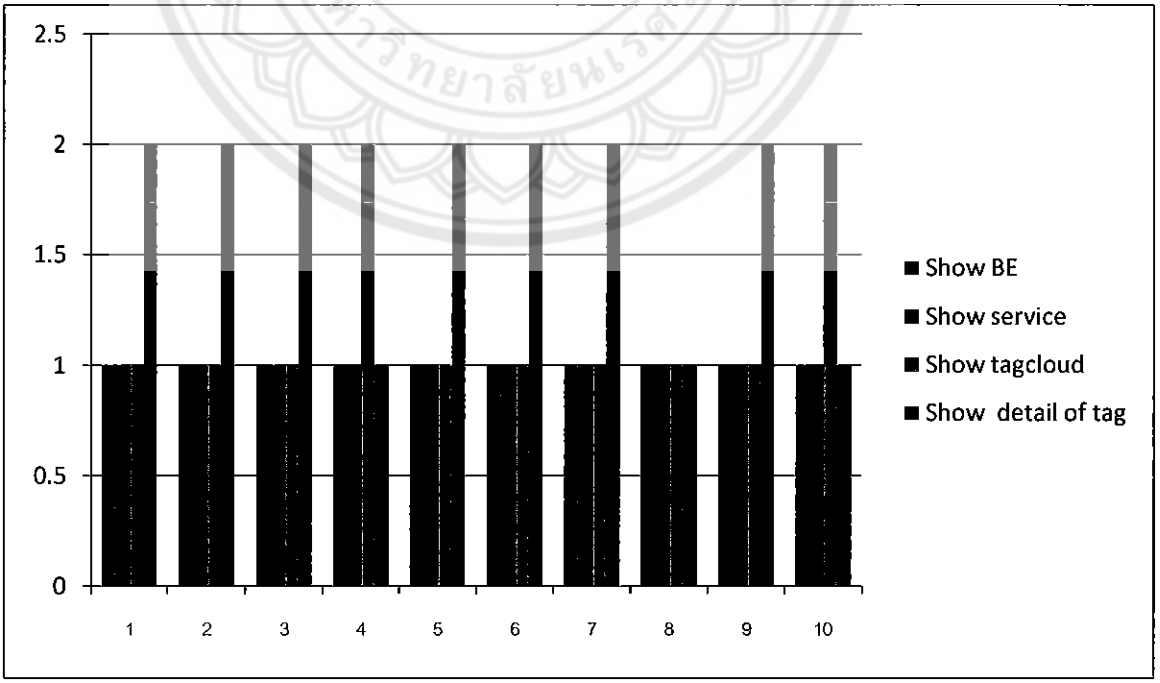
ครั้งที่	Show BE	Show web service result	Show tag cloud	Show detail of tag
1	1	1	1	2
2	1	1	1	2
3	1	1	1	2
4	1	1	2	1
5	1	1	1	2
6	1	1	1	2
7	1	1	1	2
8	1	1	1	1
9	1	1	1	2
10	1	1	2	1
Average	1	1	1.2	1.7

จากตารางที่ 5.6 การสร้างและแสดงนิพจน์ทางตรรกศาสตร์ (Boolean Expressions) ใช้เวลา 1 วินาที การแสดงผลการค้นหาเว็บเซอร์วิสใช้เวลา 1 วินาที การแสดง tag cloud ใช้เวลา 1.2 วินาที การแสดงรายละเอียดของแท็กเมื่อผู้ใช้คลิ้กใช้เวลา 1.7 วินาที โดยประมาณ จากการทดสอบ 10 ครั้ง

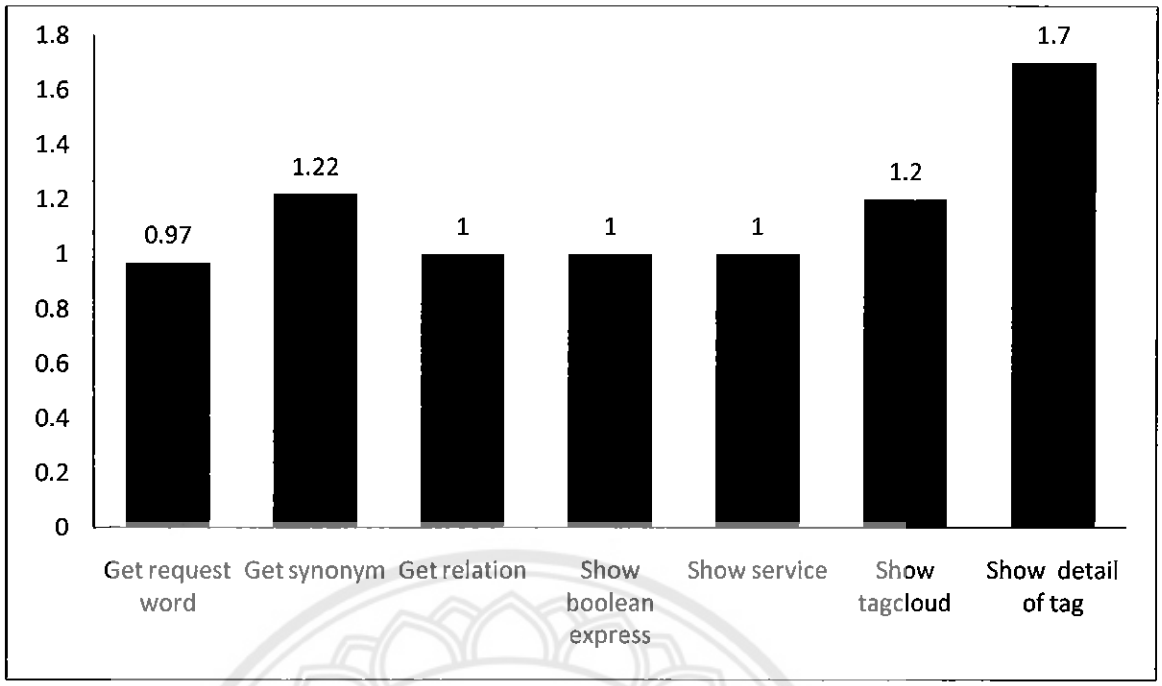
5.2.5 กราฟแสดงการทดสอบเวลาการทำงานของระบบต่าง ๆ



รูปที่ 5.3 แสดงกราฟการทดสอบเวลาการทำงานของระบบ



รูปที่ 5.4 แสดงกราฟการทดสอบเวลาการทำงานของระบบในขั้นตอนของการแสดงผลต่างๆ



รูปที่ 5.5 แสดงกราฟการทดสอบเวลาการทำงานของระบบทั้งหมด

บทที่ 6
บทสรุปและข้อเสนอแนะ

6.1 สรุปผลการดำเนินงาน

โครงการนี้พัฒนาขึ้นเพื่อศึกษาและสร้างส่วนต่อประสานระบบค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟ โดยได้ทำการสร้างเว็บแอบพลิเคชั่น ซึ่งเป็นอินเตอร์เฟส (interface) ที่ติดต่อการทำงานระหว่างระบบกับผู้ใช้ โดยมีเป้าหมายสำคัญคือค้นหาข้อมูลที่ต้องการแล้วแสดงผลการค้นหาซึ่งตรงตามความหมายที่ผู้ใช้งานต้องการ ระบบที่สร้างขึ้นสามารถเลือกคำนามจากประโยคค้นหา นำคำนามดังกล่าวมาค้นหาคำที่มีความหมายเหมือนกัน และปรับปรุงประโยคที่ผู้ใช้ต้องการค้นหาให้สอดคล้องกับ ontology แล้วแสดงผลผ่านส่วนต่อประสานดังกล่าว ซึ่งผู้ใช้สามารถเลือกคำหรือประโยคแนะนำและระบบจะตอบสนองต่อความเปลี่ยนตลอดเวลา และแสดงผลการค้นหาแบบอินเตอร์แอคทีฟ นอกจากนี้ระบบยังนำกลุ่มของเว็บเซอร์วิสที่ค้นหาได้ในแต่ละครั้งที่มีการตอบสนองต่อระบบไปค้นหาแท็ก แล้วนำมาแสดงเป็น tag cloud ทำให้ผู้ใช้มีทางเลือกในการค้นหาข้อมูลอีกทางหนึ่ง หากผู้ใช้คลิ้กแท็ก ระบบจะแสดงกลุ่มของเว็บเซอร์วิสที่เกียวข้องกับแท็กดังกล่าว จากการทำงานทั้งหมดทำให้การค้นหาข้อมูลได้ผลการค้นหาที่ตรงต่อความต้องการของผู้ใช้มากขึ้นและยังแนะนำคำค้นหาอื่นๆที่เกี่ยวข้องให้ผู้ใช้มีตัวเลือกในการค้นหาอีกด้วย

6.2 ปัญหาและอุปสรรคที่พบในการดำเนินงาน

6.2.1 เนื่องจากโครงงานระบบการค้นหาเว็บเซอร์วิสแบบอินเตอร์แอคทีฟมีผู้ร่วมโครงงานทั้งหมด 3 กลุ่ม การทำงานจึงเชื่อมต่อกันของแต่ละส่วน ซึ่งมีการแบ่งการทำงานออกเป็นส่วนๆ ที่ชัดเจน การทำงานของโปรแกรมจึงต้องมีการเชื่อมต่อส่วนต่างๆเข้าด้วยกัน ทำให้เกิดปัญหาในขั้นตอนดังกล่าว เช่น การเรียกใช้เว็บเซอร์วิส รูปแบบไฟล์ข้อมูลเข้าและไฟล์ข้อมูลออก (input and output format) เป็นต้น

6.2.2 การทำงานในส่วนของการประมวลผล เนื้อหาคำที่มีความหมายเหมือนกัน(Synonym) นั้นใช้เวลามาก ทำให้ระบบตอบสนองช้ากว่าส่วนอื่นๆมาก โครงงานนี้ใช้ wordNet API ในการประมวลผลเพื่อหา synonym ซึ่งจะเรียกใช้ไฟล์ไดนามิก ลิงค์ ไลบรารี่ Dynamic Link Library หรือไฟล์ .dll

6.2.3 ระบบที่สร้างขึ้นจะค้นหาความสัมพันธ์ของคำที่ต้องการค้นหา ที่ผู้ใช้เลือกจาก ontology ซึ่งถูกอธิบายด้วย ontology web Languages จากการทดลองปฏิบัติจริง พบว่าไฟล์ที่นำมาใช้นั้นมีรูปแบบการอธิบายที่แตกต่างกัน ทำให้เกิดความไม่ยืดหยุ่นในบางกรณี

6.3 ข้อเสนอแนะและแนวทางแก้ไข

6.3.1 ผู้ร่วมโครงงานต้องกำหนดรูปแบบข้อมูลขาเข้าและข้อมูลขาออกของระบบของตนเองอย่างชัดเจน และประสานงานกับผู้ร่วมโครงงานกลุ่มอื่นอย่างสม่ำเสมอ นอกจากนี้ต้องชี้แจงความคืบหน้าและความเปลี่ยนแปลงของระบบทุกครั้ง หากการเปลี่ยนแปลงดังกล่าวส่งผลต่อระบบอื่น

6.3.2 ศึกษา API ที่ใช้หาคำที่มีความหมายเหมือนกัน ชี้แจงรายละเอียดส่วนงานของตนเองให้ผู้ร่วมงานทุกครั้งที่มีการเปลี่ยนแปลง ซึ่งมีผลกระทบต่อส่วนอื่นๆ และเปรียบเทียบประสิทธิภาพการทำงาน

6.3.3 เขียนโปรแกรมเพื่อหาความสัมพันธ์ซึ่งมีความครอบคลุมกับรูปแบบในไฟล์ owl มากขึ้น

เอกสารอ้างอิง

[1] Webmaster. **"Mike's Digital Laboratory: semantic web technologies"** [Online]. Available : http://www.mikeaxelrod.com/wp/tag/semantic-web-technologies. 2009.

[2] Yoo Jung An. **"ONTOLOGY LEARNING FOR THE SEMANTIC DEEP WEB"**. พิมพ์ครั้งที่ 1. New Jersey Institute of Technology Newark, NJ, USA. 2008.

[3] Webmaster. **"Java API for WordNet Searching (JAWS)"** [Online]. Available: http://lyle.smu.edu/~tspell/jaws/index.html. 2009.

[4] Adwait Ratnaparkhi. **"A Maximum Entropy Model for Part-Of-Speech Tagging"**. University of Pennsylvania.

[5] Sarah Huggett. **"Social networking in academia"** Researchtrends. Issue 16 March 2010.pp.5-6

[6] Lichan Hong, Gregorio Convertino, Bongwon Suh, Ed H. Chi, and Sanjay Kairam. **"FeedWinnower: Layering Structures Over Collections of Information Streams"**. Palo Alto Research Center(PARC)

[7] Satnam Alag. **"Collective Intelligence in Action"** : Manning Publication.October 28,2008.pp.50-81

[8] Webmaster. **"Mike's Stanford Log-linear Part-Of-Speech Tagger"** [Online]. Available : http://nlp.stanford.edu/software/tagger.shtml. 2003.

[9] Webmaster. **"Part-of-speech tagging"** [Online]. Available : http://en.wikipedia.org/wiki/Part-of-speech_tagging. 2011.

[10] Webmaster. **"How To: Execute command line in C#, get STD OUT results"** [Online]. Available : http://stackoverflow.com/questions/206323/how-to-execute-command-line-in-c-get-std-out-results. 2011.

[11] Webmaster. **"Searches Per Day"** [Online]. Available : http://searchenginewatch.com/2156461. 2006.

[12] Webmaster. **"comScore Releases December 2010 U.S. Search Engine Rankings"** [Online]. Available : http://www.comscore.com/Press_Events/Press_Releases/2011/1/comScore_Releases_December_2010_U.S._Search_Engine_Rankings. 2010.

[13] Webmaster. **"OWL Web Ontology Language Overview"** [Online]. Available : http://www.w3.org/TR/owl-features. 2011.

เอกสารอ้างอิง(ต่อ)

[14] Webmaster. **"WordNet A lexical database for English"** [Online]. Available: http://wordnet.princeton.edu/wordnet/related-projects. 2011.

[15] Webmaster. **"Revision 6115: /Source/ResourceAPIs/WordNet"** [Online]. Available : http://links.cse.msu.edu:8000/svn/NLP/Source/ResourceAPIs/WordNet.

[16] Webmaster. **"SCM Repositories - nhunspell"** [Online]. Available : http://nhunspell.svn.sourceforge.net/viewvc/nhunspell. 2011.

[17] Webmaster. **"NHunspell: Spellcheck, Hyphen, Thesaurus"** [Online]. Available : http://sourceforge.net/projects/nhunspell. 2011.

[18] Webmaster. **"OWL-S Service Retreival Test Collection:File Release Notes and Changelog"** [Online]. Available : http://www.semwebcentral.org/frs/shownotes.php?release_id=369.

[19] Webmaster. **"OWL-S Service Retreival Test Collection: Project Filelist"** [Online]. Available : http://www.semwebcentral.org/frs/?group_id=89.

[20] Webmaster. **"OWL"** [Online]. Available : http://en.wikipedia.org/wiki/Owl.

[21] Webmaster. **"Resource Description Framework (RDF)"** [Online]. Available : http://www.w3.org/RDF. 2004.

[22] Webmaster. **"2007 NAICS Codes"** [Online]. Available : http://www.census.gov/naics/2007/NAICOD07.HTM. 2007.

[23] Webmaster. **"North American Industry Classification System"** [Online]. Available : http://www.census.gov/eos/www/naics. 2007.

[24] Webmaster. **"What is Boolean Searching"** [Online]. Available : http://www.wallhowe.com/navnet/search2.html. 2000.

[25] Webmaster. "Automatic query expansion using SMART: Trec-3 report, Third text retrieval conference, 1995

[26] Automatic feedback using past queries: social searching? ACM SIGIR'97

[27] Query expansion using associated queries, CIKM'03

[28] Relevant term suggestion in interactive web search based on contextual information in query session log, JASIST, 54(7), 2003

[29] Concept-based query expansion, ACM SIGIR 1993

[30] Concept-based interactive query expansion, ACM SIGIR 2005

เอกสารอ้างอิง(ต่อ)

[31] Incremental formalization of document annotations through ontology-based paraphrasing, WWW2004

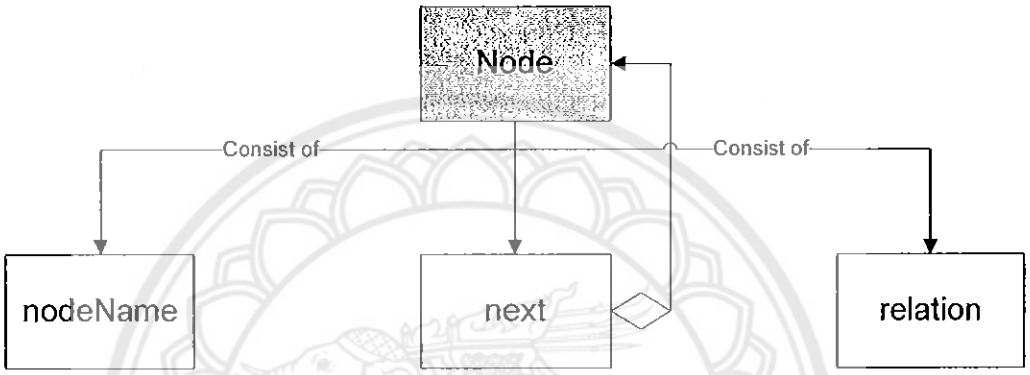
[32] Hierarchical presentation of expansion terms, ACM SAC 2002

[33] Query Reformulation on the Internet: Empirical Data and the Hyper index Search Engine,RIA097

[34] Webmaster. "**My Delicious Tags**" [Online]. Available : http://www.delicious.com/help/tagrolls

ภาคผนวก ก

ตัวอย่างโค้ดการสร้างแผนภูมิต้นไม้ (Tree) ของความสัมพันธ์ภายในออนโทโลยีได้ และนำแผนภูมิต้นไม้ดังกล่าวไปสร้างเป็นประโยคแนะนำให้แก่ผู้ใช้ ทำให้ประโยคที่ได้มีความสอดคล้องกับออนโทโลยี เมื่อผู้ใช้เลือกประโยคแนะนำจะทำให้การค้นหาตรงกับข้อมูลที่ระบบมีอยู่มากขึ้น และประสิทธิภาพในการค้นหาจะเพิ่มมากยิ่งขึ้น

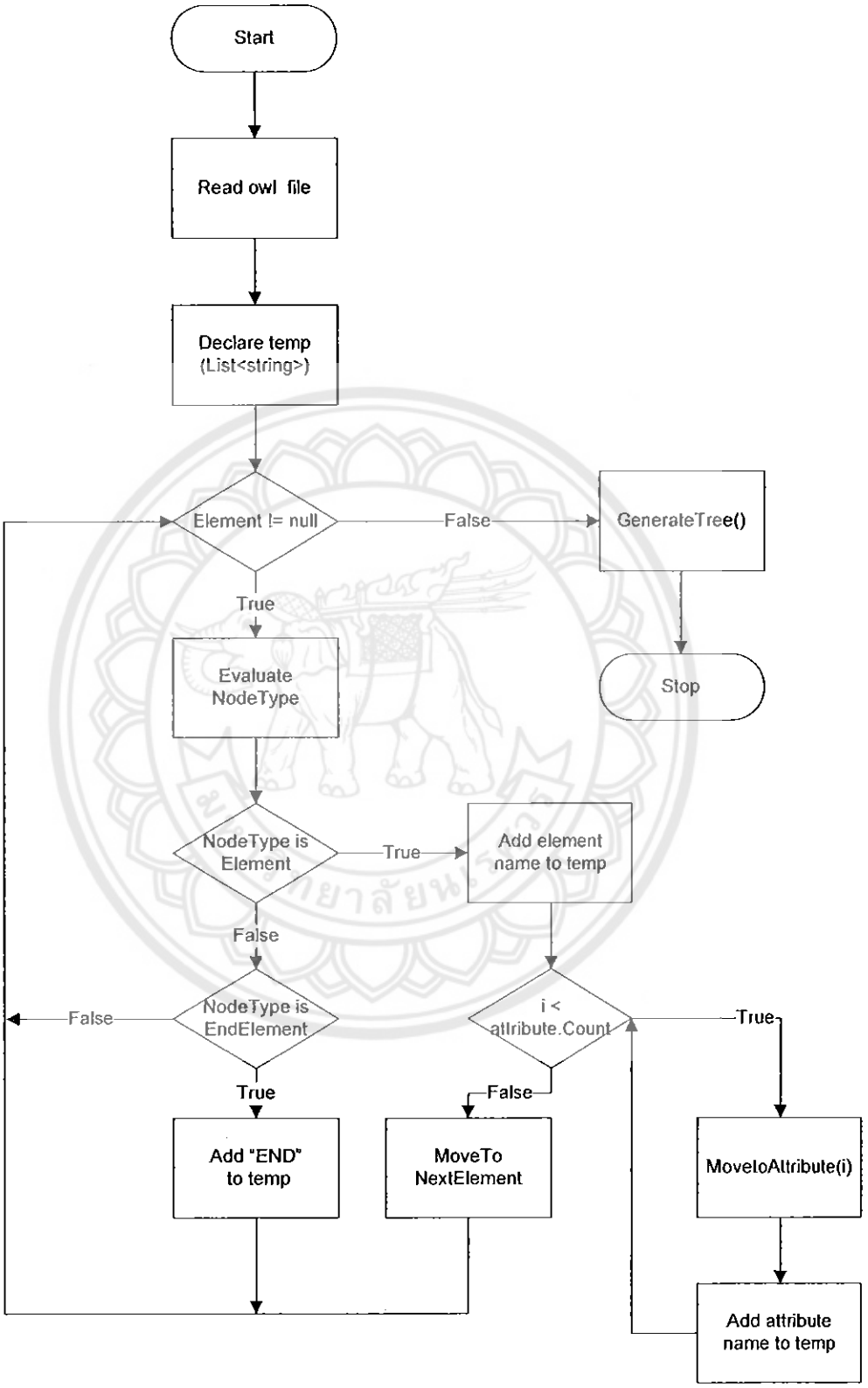


รูปที่ ก1 แสดงโครงสร้างของโหนด (Node)

```
public class Node
{
    public string nodeName;
    public List<Node> next;
    public List<string> relation;

    public Node(string NodeName)
    {
        this.nodeName = NodeName;
        this.next = new List<Node>();
        this.relation = new List<string>();
    }
}
```

รูปที่ ก2 แสดงตัวอย่างโค้ดที่กำหนดโครงสร้างของโหนด (Node)

การ parse owl file และเก็บข้อมูลเพื่อนำมาสร้างแผนภูมิต้นไม้ของ ontology

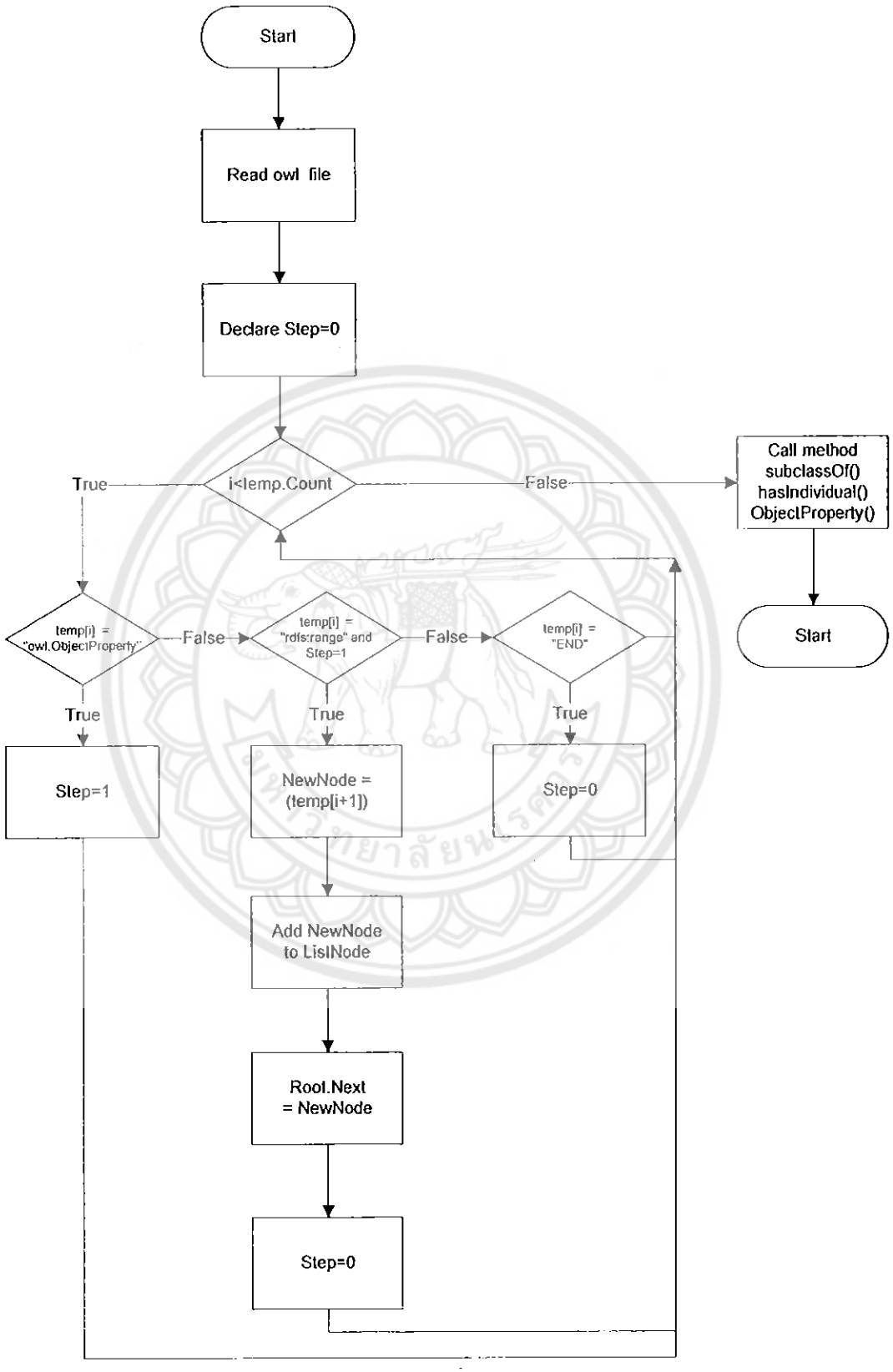


รูปที่ ก3 แสดงแผนผังการทำงานของโปรแกรม parser owl file

```
public void Parser(string FileName)
   {
       try
       {
           XmlTextReader reader = new XmlTextReader(FileName);
           while (reader.Read())
           {
               switch (reader.NodeType)
               {
                   case XmlNodeType.Element:
                       temp.Add(reader.Name);
                       for (int i = 0; i < reader.AttributeCount; i++)
                       {
                           reader.MoveToAttribute(i);

                           if (reader.Name.Equals("rdf:ID") ||
                           reader.Name.Equals("rdf:about") ||
                           reader.Name.Equals("rdf:resource"))
                           {
                               temp.Add(reader.Value.ToLower());
                           }
                       }
                       reader.MoveToElement();
                       break;

                   case XmlNodeType.EndElement:
                       if (reader.Name.Equals("owl:Class"))
                       {
                           temp.Add("END");
                       }
                       break;
                   default: break;
               }
           }
           GenerateTree();
       }
       catch (Exception ex)
       {
           Console.WriteLine(ex.Message);
       }
   }
```

รูปที่ ก4 แสดงแผนผังการทำงานของโปรแกรม parser owl file

การสร้างแผนภูมิต้นไม้โดยออนโทโลยี



รูปที่ ก5 แสดงแผนผังการทำงานของโปรแกรมส่วนการสร้างแผนภูมิต้นไม้โดยออนโทโลยี

```
public void GenerateTree()
   {
       int step = 0;

       for (int i = 0; i < temp.Count; i++)
       {
           if (temp[i].Equals("owl:ObjectProperty"))
           {
               step = 1;
           }
           else if (temp[i].Equals("rdfs:range") && step == 1)
           {
               if (checkExist(temp[i + 1]))
               {
               }
               else
               {
                   Node node = new Node(delCharp(temp[i + 1]));
                   ListNode.Add(node);
                   root.next.Add(node);
                   step = 0;
               }
           }
           else if (temp[i].Equals("END"))
           {
               step = 0;
           }
       }

       subclassOf();
       hasIndividual();
       ObjectProperty();
       showNodePreOrder();
   }
```

รูปที่ ก6 แสดงตัวอย่างโค้ดในการสร้างแผนภูมิต้นไม้โดยออนโทโลยี

การดึงความสัมพันธ์แบบ subclassOf จาก ออนโทโลยี

```
  public void subclassOf()
        {
            string buff;
            for (int i = 0; i < temp.Count; i++)
            {
                if (temp[i].Equals("rdfs:subClassOf"))
                {
                    Node A = new Node("");
                    Node B = new Node("");

                    int k = i;

                    while ((!temp[k - 1].Equals("owl:Class") ||
temp[k][0].Equals('#')) && (!temp[k - 2].Equals("END")))
                    {
                        k--;
                    }

                    if (checkExist(delCharp(temp[k])))
                    {
                        for (int j = 0; j < ListNode.Count; j++)
                        {
                            if
(ListNode[j].nodeName.Equals(delCharp(temp[k])))
                            {
                                A = ListNode[j];
                            }
                        }
                    }
                    else
                    {
                        A = new Node(delCharp(temp[k]));
                        ListNode.Add(A);
                    }

                    if (temp[i + 1].Equals("owl:Class"))
                    {
                        buff = temp[i + 2];
                    }
                    else if (temp[i + 1][0] == '#')
                    {
                        buff = temp[i + 1];
                    }
                    else
                    {
                        buff = "Empty";
                    }

                    if (!buff.Equals("Empty"))
                    {
                        if (checkExist(delCharp(buff)))
                        {
                            for (int j = 0; j < ListNode.Count; j++)
                            {
                                if
(ListNode[j].nodeName.Equals(delCharp(buff)))
                                {
                                    B = ListNode[j];
                                    B.next.Add(A);
                                    B.relation.Add("is subclass of");
                                }
                            }
                        }
```

```
                        else
                        {
                            B = new Node(delCharp(buff));
                            ListNode.Add(B);
                            B.next.Add(A);
                            B.relation.Add("is subclass of");

                        }
                    }
                }

  if (temp[i].Equals("owl:equivalentClass"))
                {
                    Node A = new Node("");
                    Node B = new Node("");

                    int k = i;

                    while ((!temp[k - 1].Equals("owl:Class") ||
temp[k][0].Equals('#')) && (!temp[k - 2].Equals("END")))
                    {
                        k--;
                    }

                    if (checkExist(delCharp(temp[k])))
                    {
                        for (int j = 0; j < ListNode.Count; j++)
                        {
                            if
(ListNode[j].nodeName.Equals(delCharp(temp[k])))
                            {
                                A = ListNode[j];
                            }
                        }
                    }
                    else
                    {
                        A = new Node(delCharp(temp[k]));
                        ListNode.Add(A);
                    }

                    if ((temp[i + 1].Equals("owl:Class") && temp[i +
2].Equals("owl:intersectionOf")))
                    {
                        buff = temp[i + 4];
                    }
                    else if (temp[i + 4][0] == '#')
                    {
                        buff = temp[i + 4];
                    }

                    else
                    {
                        buff = "Empty";
                    }
```

```
if (!buff.Equals("Empty"))
                    {
                        if (checkExist(delCharp(buff)))
                        {
                            for (int j = 0; j < ListNode.Count; j++)
                            {
                                if
(ListNode[j].nodeName.Equals(delCharp(buff)))
                                {
                                    B = ListNode[j];
                                    B.next.Add(A);
                                    B.relation.Add("is subclass of");
                                }
                            }
                        }
                        else
                        {
                            B = new Node(delCharp(buff));
                            ListNode.Add(B);
                            B.next.Add(A);
                            B.relation.Add("is subclass of");

                        }
                    }
                }
            }
        }
```

รูปที่ ก7 แสดงตัวอย่างโค้ดการดึงความสัมพันธ์แบบ subclassOf จาก ออนโทโลยี

การดึงความสัมพันธ์แบบ hasIndividual จาก ออนโทโลยี

```
public void hasIndividual()
        {
            string buff;
            for (int i = 0; i < temp.Count; i++)
            {
                if (temp[i].Equals("rdfs:hasIndividual"))
                {
                    Node A = new Node("");
                    Node B = new Node("");

                    int k = i;

                    while ((!temp[k - 1].Equals("owl:Class") ||
temp[k][0].Equals('#')) && (!temp[k - 2].Equals("END")))
                    {
                        k--;
                    }

                    if (checkExist(delCharp(temp[k])))
                    {
                        for (int j = 0; j < ListNode.Count; j++)
                        {
                            if
(ListNode[j].nodeName.Equals(delCharp(temp[k])))
                            {
                                A = ListNode[j];
                            }
                        }
                    }
                    else
                    {
                        A = new Node(delCharp(temp[k]));
                        ListNode.Add(A);
                    }

                    if (temp[i + 1].Equals("owl:Class"))
                    {
                        buff = temp[i + 2];
                    }
                    else if (temp[i + 1][0] == '#')
                    {
                        buff = temp[i + 1];
                    }
                    else
                    {
                        buff = "Empty";
                    }

                    if (!buff.Equals("Empty"))
                    {
                        if (checkExist(delCharp(buff)))
                        {
                            for (int j = 0; j < ListNode.Count; j++)
                            {
                            B.relation.Add("is a ");
```

```
      if (ListNode[j].nodeName.Equals(delCharp(buff)))
                                {
                                    B = ListNode[j];
                                    B.next.Add(A);
                                    B.relation.Add("is the ");
                                }
                            }
                        }
                        else
                        {
                            B = new Node(delCharp(buff));
                            ListNode.Add(B);
                            B.next.Add(A);
                            B.relation.Add("is a ");

                        }
                    }
                }
            }
        }
```

รูปที่ ก8 แสดงตัวอย่างโค้ดการดึงความสัมพันธ์แบบ hasIndividual จาก ออนโทโลยี

การดึงความสัมพันธ์แบบ ObjectProperty จาก ออนโทโลยี

```
    public void ObjectProperty()
        {
            string buff;
            for (int i = 0; i < temp.Count; i++)
            {
                if (temp[i].Equals("owl:onProperty"))
                {
                    Node A = new Node("");
                    Node B = new Node("");

                    int k = i;

                    while ((!temp[k - 1].Equals("owl:Class") ||
temp[k][0].Equals('#')) && (!temp[k - 2].Equals("END")))
                    {
                        k--;
                    }

                    if (checkExist(delCharp(temp[k])))
                    {
                        for (int j = 0; j < ListNode.Count; j++)
                        {
                            if
(ListNode[j].nodeName.Equals(delCharp(temp[k])))
                            {
                                A = ListNode[j];

                            }
                        }
                    }
```

```
                    else
                    {


                        A = new Node(delCharp(temp[k]));
                        ListNode.Add(A);
                    }

                    if (temp[i - 2].Equals("owl:someValuesFrom") || temp[i -
2].Equals("owl:hasValue"))
                    {
                        buff = temp[i - 1];

                    }
                    else
                    {
                        buff = "Empty";
                    }

                    if (!buff.Equals("Empty"))
                    {
                        if (checkExist(delCharp(buff)))
                        {
                            for (int j = 0; j < ListNode.Count; j++)
                            {
                                if
(ListNode[j].nodeName.Equals(delCharp(buff)))
                                {
                                    B = ListNode[j];
                                    A.next.Add(B);
                                    A.relation.Add(temp[i + 2]);
                                    int z = 0;
                                }
                            }
                        }
                        else
                        {
                            B = new Node(delCharp(buff));
                            ListNode.Add(B);
                            A.next.Add(B);
                            A.relation.Add(delCharp(temp[i + 2]));

                        }
                    }
                }
            }
        }
```

รูปที่ ก9 แสดงตัวอย่างโค้ดการดึงความสัมพันธ์แบบ ObjectProperty จาก ออนโทโลยี

การค้นหาโหนดที่เป็น Specialization จากแผนภูมิต้นไม้ของออนโทโลยี

```
public List<string> getSpecialization()
{
       List<string> result = new List<string>();
       for (int i = 0; i < ListNode.Count; i++)
       {
           if (ListNode[i].nodeName.Equals(this.InputNode))
           {
               for (int j = 0; j < ListNode[i].next.Count; j++)
               {
                   if ((ListNode[i].relation[j][0] != '#'))
                   {
                    Console.WriteLine(ListNode[i].next[j].nodeName + "   "
                     + ListNode[i].relation[j] + "   " + ListNode[i].nodeName);
                    result.Add("$"+ListNode[i].next[j].nodeName + "   " +
                    ListNode[i].relation[j] + "   " + istNode[i].nodeName);
                   }
               }
           }
       }
       return result;
}
```

รูปที่ ก10 แสดงตัวอย่างโค้ดการค้นหาโหนดที่เป็น Specialization จากแผนภูมิต้นไม้ของออนโทโลยี

การค้นหาโหนดที่เป็น Generalization จากแผนภูมิต้นไม้ของออนโทโลยี

```
public List<string> getGeneralization()
{
        List<string> result = new List<string>();

        for (int i = 0; i < ListNode.Count; i++)
        {
            for (int j = 0; j < ListNode[i].next.Count; j++)
            {
                if (ListNode[i].next[j].nodeName.Equals(this.InputNode))
                {
                    if ((ListNode[i].relation[j][0] != '#'))
                    {
                    Console.WriteLine(ListNode[i].next[j].nodeName + "   " +
                    ListNode[i].relation[j] + "   " + ListNode[i].nodeName);
                    result.Add(ListNode[i].next[j].nodeName + "   " +
                    ListNode[i].relation[j] + "  $" + ListNode[i].nodeName);
                    }
                }
            }
        }
        return result;
}
```

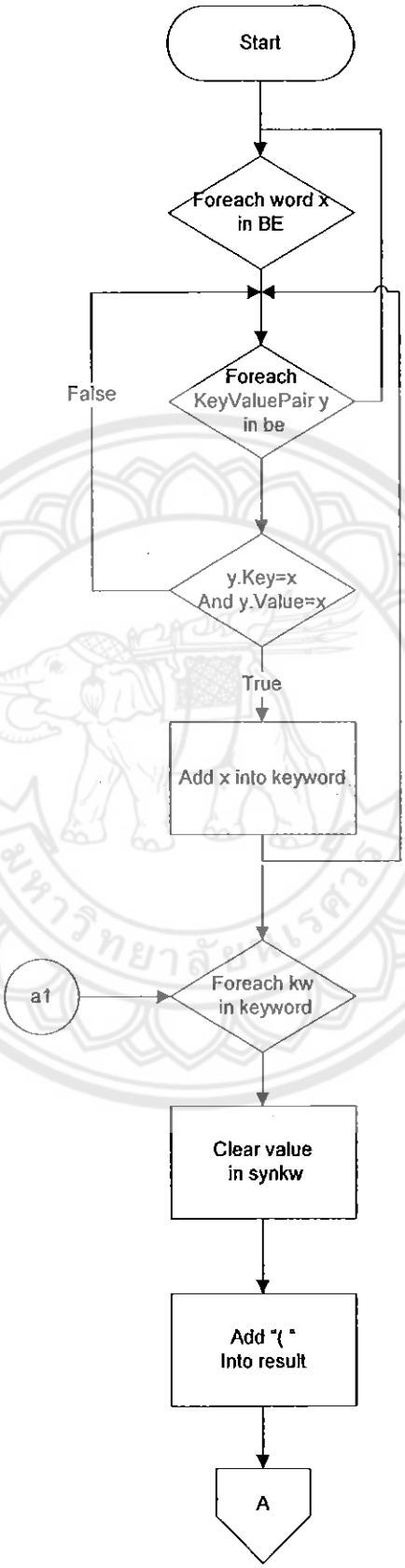
รูปที่ ก11 แสดงตัวอย่างโค้ดการค้นหาโหนดที่เป็น Generalization จากแผนภูมิต้นไม้ของออนโทโลยี

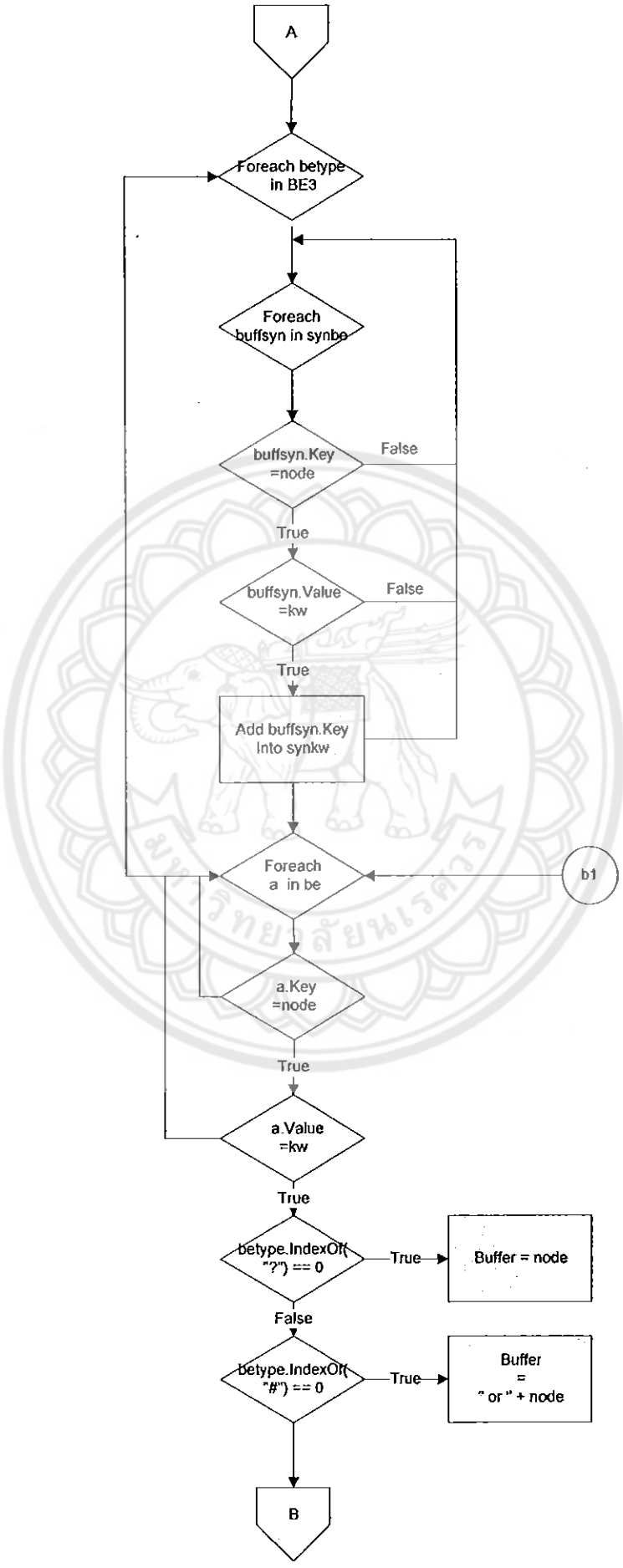
การค้นหาโหนดที่เป็น Association จากแผนภูมิต้นไม้ของออนโทโลยี

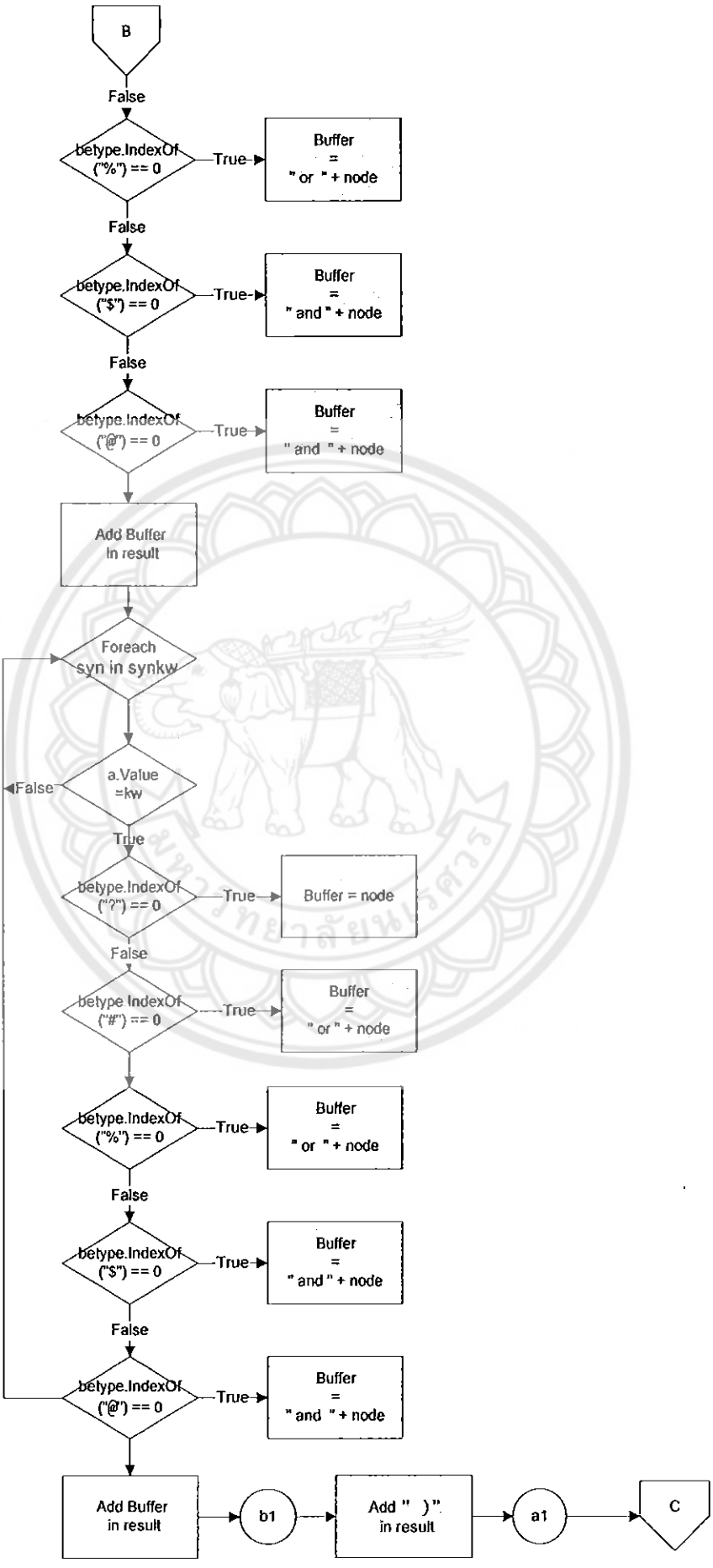
```
public List<string>  getAssociation()
{
       List<string> result = new List<string>();
       for (int i = 0; i < ListNode.Count; i++)
       {
           for (int j = 0; j < ListNode[i].next.Count; j++)
           {
               if (ListNode[i].next[j].nodeName.Equals(this.InputNode))
               {
                   if ((ListNode[i].relation[j][0] == '#'))
                   {
                   Console.WriteLine(ListNode[i].next[j].nodeName + "   " +
                   delCharp(ListNode[i].relation[j]) + " at " + "  " +
                   ListNode[i].nodeName);
                   result.Add(ListNode[i].next[j].nodeName + "   " +
                   delCharp(ListNode[i].relation[j]) + " at " + "  $" +
                   ListNode[i].nodeName);
                   }
               }
               if (ListNode[i].nodeName.Equals(this.InputNode))
               {
                   if ((ListNode[i].relation[j][0] == '#'))
                   {
                   Console.WriteLine(ListNode[i].nodeName + "   " +
                   delCharp(ListNode[i].relation[j]) + "   " +
                   ListNode[i].next[j].nodeName);
                   result.Add(ListNode[i].nodeName + "   " +
                   delCharp(ListNode[i].relation[j]) + "   " +
                   ListNode[i].next[j].nodeName);
                   }
               }
           }
       }
       return result;
}
```

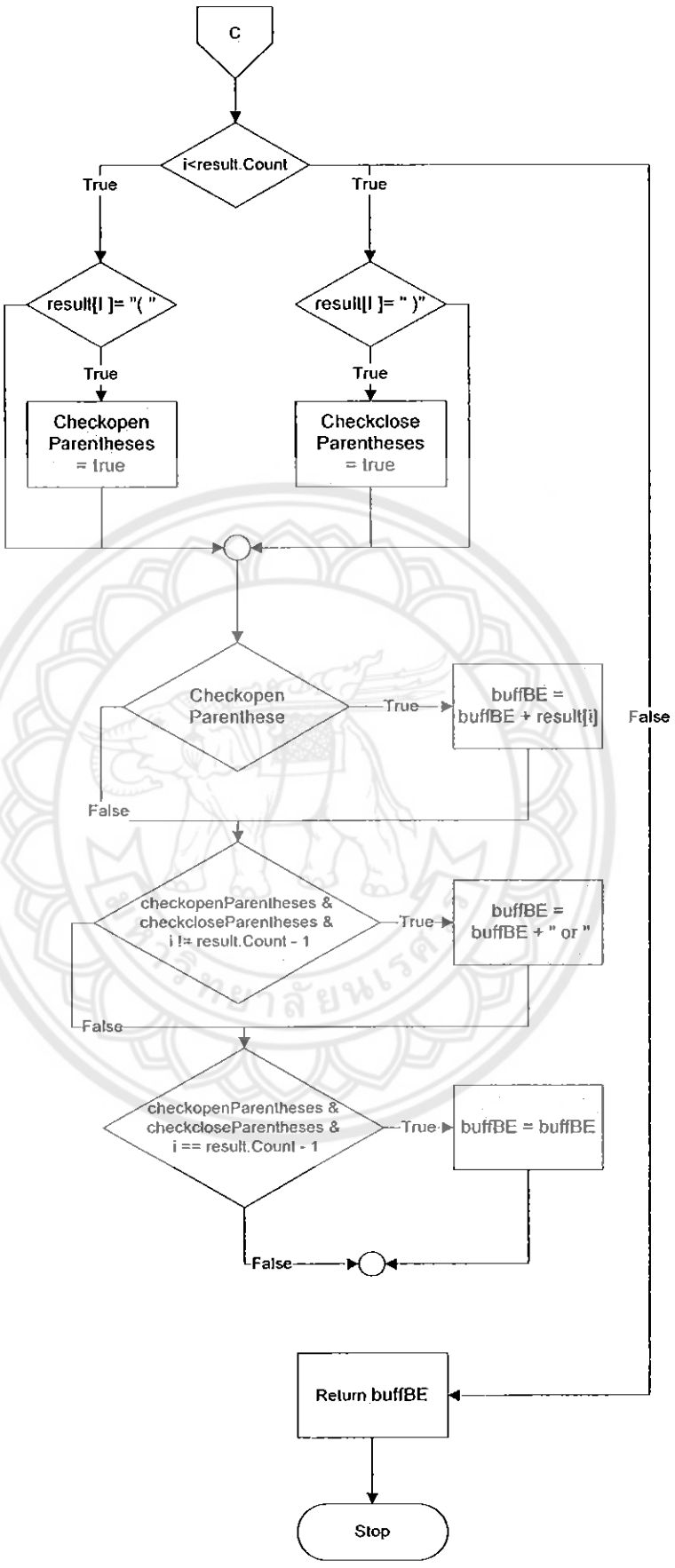
รูปที่ ก12 แสดงตัวอย่างโค้ดการค้นหาโหนดที่เป็น Association จากแผนภูมิต้นไม้ของออนโทโลยี

การสร้างนิพจน์ทางตรรกศาสตร์ (Boolean Expression) จากคำที่ผู้ใช้เลือก




A
Foreach betype in BE3
Foreach buffsyn in synbe
buffsyn.Key =node
False
True
buffsyn.Value =kw
False
True
Add buffsyn.Key Into synkw
Foreach a in be
b1
a.Key =node
True
a.Value =kw
True
betype.IndexOf("?") == 0
True
Buffer = node
False
betype.IndexOf("#") == 0
True
Buffer = " or " + node
B


B
False
betype.IndexOf ("%") == 0
True
Buffer = " or " + node
False
betype.IndexOf ("$") == 0
True
Buffer = " and " + node
False
betype.IndexOf ("@") == 0
True
Buffer = " and " + node
Add Buffer in result
Foreach syn in synkw
a.Value =kw
False
True
betype.IndexOf ("?") == 0
True
Buffer = node
False
betype.IndexOf ("#") == 0
True
Buffer = " or " + node
betype.IndexOf ("%") == 0
True
Buffer = " or " + node
False
betype.IndexOf ("$") == 0
True
Buffer = " and " + node
False
betype.IndexOf ("@") == 0
True
Buffer = " and " + node
Add Buffer in result
b1
Add ") " in result
a1
C



รูปที่ ก13 แสดงตัวอย่างแผนผังการทำงานของโปรแกรมสร้างนิพจน์ทางตรรกศาสตร์

```
public string showBooleanExpression()
{
    BooleanExpressionTextBox.Text = string.Empty;
    string buff = "";
    string buffBE = "";
    string node = "";

    List<string> keyword = new List<string>();
    foreach (string x in BE)
    {
        foreach (KeyValuePair<string, string> a in be)
        {
            if (a.Key.Equals(x) && a.Value.Equals(x))
            {
                keyword.Add(a.Key);
            }
        }
    }
    List<string> result = new List<string>();
    List<string> synkw = new List<string>();
    foreach (string kw in keyword)
    {
         synkw.Clear();
         result.Add("( ");
         foreach (string betype in BE3)
         {
             node = betype.Remove(0, 1);
             foreach (KeyValuePair<string, string> buffsyn in synbe)
             {
                 if (buffsyn.Key.Equals(node))
                 {
                     if (buffsyn.Value.Equals(kw))
                     {
                         if (!checkExist(synkw, buffsyn.Key))
                         synkw.Add(buffsyn.Key);
                     }
                 }
             }
```

```
foreach (KeyValuePair<string, string> a in be)
{
    if (a.Key.Equals(node))
    {
         if (a.Value.Equals(kw))
         {
              if (betype.IndexOf("?") == 0)
              {
                    buff = node;
              }
              else if (betype.IndexOf("#") == 0)
              {
                    buff = " or " + node;
              }
              else if (betype.IndexOf("%") == 0)
              {
                    buff = " or " + node;
              }
              else if (betype.IndexOf("$") == 0)
              {
                    buff = " and " + node;
              }
              else if (betype.IndexOf("@") == 0)
              {
                    buff = " and " + node;
              }
              else
              {
              }
              result.Add(buff);
         }
```

```
foreach (string syn in synkw)
{
        if (a.Value.Equals(syn))
        {
             if (betype.IndexOf("?") == 0)
             {
                  buff = " " + node;
             }
             else if (betype.IndexOf("#") == 0)
             {
                  buff = " or " + node;
             }
             else if (betype.IndexOf("%") == 0)
             {
                  buff = " or " + node;
             }
             else if (betype.IndexOf("$") == 0)
             {
                  buff = " and " + node;
             }
             else if (betype.IndexOf("@") == 0)
             {
                  buff = " and " + node;
             }
             else
             {
             }
             result.Add(buff);
        }
       }
     }
    }
 }
result.Add(" )");
}
```

```
bool checkcloseParentheses = false;
bool checkopenParentheses = false;

for (int i = 0; i < result.Count; i++) // Loop through List with for
{
      if (result[i].Equals("( "))
      {
           checkopenParentheses = true;
      }
      if (result[i].Equals(" )"))
      {
           checkcloseParentheses = true;
      }
      if (checkopenParentheses)
      {
           buffBE = buffBE + result[i];
      }
      if (checkopenParentheses&&checkcloseParentheses&&i!=result.Count-1)
      {
           buffBE = buffBE + " or ";
           checkcloseParentheses = false;
           checkopenParentheses = false;
      }
      if (checkopenParentheses&&checkcloseParentheses&&i==result.Count-1)
      {
           buffBE = buffBE;
           checkcloseParentheses = false;
           checkopenParentheses = false;
      }
}
BooleanExpressionTextBox.Text = buffBE;

return buffBE;
}
```

รูปที่ ก14 แสดงตัวอย่างโค้ดของโปรแกรมสร้างนิพจน์ทางตรรกศาสตร์

ภาคผนวก ข

เนื่องจากระบบที่พัฒนาขึ้นมีการประยุกต์ใช้ Stanford Log-linear Part-Of-Speech Tagger เพื่อแยกประเภทของคำ ซึ่งประเภทของคำที่แสดงโดย Stanford Log-linear Part-Of-Speech Tagger จะแสดงในปแบบของ Penn Treebank Tag set ซึ่งใช้ตัวอักษรย่อแทนชื่อของชนิดของคำ ดังต่อไปนี้

ตารางที่ ข1 แสดงอักษรย่อที่ระบุแทนชนิดของคำต่างๆ ใน Penn Treebank Tag set

CC	Coordinating conjunction (e.g. and,but,or...)
CD	Cardinal Number
DT	Determiner
EX	Existential *there*
FW	Foreign Word
IN	Preposision or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List Item Marker
MD	Modal (e.g. can, could, might, may...)
NN	Noun, singular or mass
NNP	Proper Noun, singular
NNPS	Proper Noun, plural
NNS	Noun, plural
PDT	Predeterminer (e.g. all, both ... when they precede an article)
POS	Possessive Ending (e.g. Nouns ending in 's)
PRP	Personal Pronoun (e.g. I, me, you, he...)
PRP$	Possessive Pronoun (e.g. my, your, mine, yours...)
RB	Adverb

	Most words that end in -ly as well as degree words like quite, too and very
RBR	Adverb, comparative Adverbs with the comparative ending -er, with a strictly comparative meaning.
RBS	Adverb, superlative
RP	Particle
SYM	Symbol Should be used for mathematical, scientific or technical symbols
TO	*to*
UH	Interjection (e.g. uh, well, yes, my...)
VB	Verb, base form subsumes imperatives, infinitives and subjunctives
VBD	Verb, past tense includes the conditional form of the verb to be
VBG	Verb, gerund or persent participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner (e.g. which, and *that* when it is used as a relative pronoun)
WP	Wh-pronoun (e.g. what, who, whom...)
WP$	Possessive wh-pronoun
WRB	Wh-adverb (e.g. how, where why)

Punctuation Tags
#
$
"
(
)
,
.
:
``

ประวัติผู้เขียนโครงงาน

ชื่อ นางสาวสุนันท์ ธาติ

ภูมิลำเนา 55 หมู่ที่ 2 ตำบลยางตาล อำเภอโกรกพระ

จังหวัดนครสวรรค์ 60170

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนจิรประวัติวิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร จังหวัดพิษณุโลก

E-mail : **tati_sunun@hotmail.com**

ชื่อ นางสาวนภารัตน์ ปุ้มตะมะ

ภูมิลำเนา 169 หมู่ 21 ตำบลนาบ่อคำ อำเภอเมือง

จังหวัดกำแพงเพชร 62000

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนหนองกองพิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร จังหวัดพิษณุโลก

E-mail : **naparatp_oom@hotmail.com**