ระบบค้นคืนสารสนเทศภาษาไทย-อังกฤษ สำหรับคำทับศัพท์และแสดงผลลัพธ์ด้วยวิธีการจัดกลุ่มข้อมูล

นางสาวกฤษณี อริยชาญศิลป์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2545

ISBN 974-17-1982-5

CROSS LANGUAGE RETRIEVAL SEARCH ENGINE FOR THAI/ENGLISH TRANSLITERATED WORD AND RESULT USING CLUSTERING

Miss Krisanee Ariyachansin

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2002

ISBN 974-17-1982-5

หัวข้อวิทยานิพนธ์	ระบบค้นคืนสารสนเทศภาษาไทย-อังกฤษสำหรับคำทับศัพท์และ แสดงผลลัพธ์ด้วยวิธีการจัดกลุ่มข้อมูล
โดย	นางสาวกฤษณี อริยชาญศิลป์
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. บุญเสริม กิจศิริกุล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร. สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร. สมชาย ประสิทธิ์จูตระกูล)

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร. บุญเสริม กิจศิริกุล)

..... กรรมการ
(อาจารย์ ดร. ฐิต ศิริบูรณ์)

.....กรรมการ
(อาจารย์ ดร. วีระ เหมืองสิน)

นางสาวกฤษณี อริยชาญศิลป์ : ระบบค้นคืนสารสนเทศภาษาไทย-อังกฤษสำหรับคำทับศัพท์และแสดงผลลัพธ์ด้วยวิธีการจัดกลุ่มข้อมูล (CROSS LANGUAGE RETRIEVAL SEARCH ENGINE FOR THAI/ENGLISH TRANSLITERATED WORD AND RESULT USING CLUSTERING)
อ. ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร. บุญเสริม กิจศิริกุล, 74 หน้า. ISBN 974-17-1982-5.

วิทยานิพนธ์ฉบับนี้นำเสนอระบบค้นคืนสารสนเทศภาษาไทย-อังกฤษสำหรับคำทับศัพท์และแสดงผลลัพธ์ด้วยวิธีการจัดกลุ่มข้อมูลแบบซัฟฟิกทรี ซึ่งอนุญาตให้ใช้ข้อคำถามที่เป็น คำทับศัพท์ภาษาอังกฤษหรือภาษาไทยในการค้นคืนเอกสารที่มีคำหลักตรงกันในอีกภาษา

ระบบค้นคืนที่ได้สามารถทำการค้นคืนข้ามภาษาไทย-อังกฤษสำหรับคำทับศัพท์ได้โดยไม่ต้องอาศัยพจนานุกรม ผู้ใช้สามารถใส่ข้อคำถามที่เป็นภาษาใดภาษาหนึ่งให้ระบบค้นคืนทำการเข้ารหัสคำด้วยเทคนิคนิวรอน-เน็ตเวิร์ก เพื่อระบบได้นำรหัสคำไปทำการค้นคืนเอกสารที่มีคำทับศัพท์ซึ่งมีรหัสเดียวกันปรากฏอยู่ ผลลัพธ์ที่ได้จากการค้นหาจะถูกนำมาจัดกลุ่มข้อมูลแบบซัฟฟิกทรีเพื่อจัดกลุ่มของ ผลลัพธ์ให้เป็นสัดส่วนสะดวกต่อการค้นหายิ่งขึ้น

ภาควิชา .....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต .....
สาขาวิชา .....วิทยาศาสตร์คอมพิวเตอร์..... ลายมือชื่ออาจารย์ที่ปรึกษา .....
ปีการศึกษา .....2545..... ลายมือชื่ออาจารย์ที่ปรึกษาร่วม .....

4271403221 : MAJOR COMPUTER SCIENCE

KEY WORD: TRANSLITERATED WORD/CROSS LANGUAGE/ SEARCH ENGINE/ CLUSTERING

THESIS ADVISOR : ASST.PROF.BOONSERM KIJSIRIKUL. 74 pp. ISBN 974-17-1982-5.

This thesis presents a cross-language retrieval search engine for Thai-English transliterated words which gives results by suffix tree clustering. The search engine is able to retrieve documents containing either the English-to-Thai or Thai-to-English transliterated keywords.

The search engine used no dictionary,and cross-language retrieval is done by encoding each word in the query terms by neural networks and then matching the query code with codes of keywords in the index. The similar documents retrieved by the search engine are then clustered using the suffix tree clustering technique for ease of use by the user.

Department …Computer Engineering Faculty… Student's signature.....

Field of Study ……Computer Science……..… Advisor's signature …….....

Academic year ……..2002…….. Co-advisor's signature .....…….....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งของผู้ช่วยศาสตราจารย์ ดร. บุญเสริม กิจศิริกุล อาจารย์ที่ปรึกษาวิทยานิพนธ์ซึ่งท่านได้ให้คำแนะนำและข้อคิดเห็นต่าง ๆ ในการวิจัยมาด้วยดีตลอดรวมทั้งตรวจแก้วิทยานิพนธ์ฉบับนี้อย่างละเอียด

ขอขอบคุณผู้พัฒนาระบบค้นคืน htdig และผู้วิจัยงานวิจัยทุกงานที่ได้ทำการพัฒนา งานวิจัยและข้อมูลอื่น ๆ ทำให้ผู้วิจัยสามารถนำมาประกอบการพัฒนางานวิจัยชิ้นนี้ให้สำเร็จลุล่วงไปได้ ด้วยดี

ขอขอบคุณเพื่อนๆ และพี่ ๆ ที่ให้ข้อเสนอแนะและเทคนิคต่าง ๆ ในการจัดทำระบบ ค้นคืนนี้ให้สำเร็จลุล่วงและเอื้อเฟื้อเครื่องมือในการพัฒนาและทดสอบระบบ

ท้ายนี้ ผู้วิจัยใคร่ขอกราบขอบพระคุณบิดามารดาซึ่งให้การเลี้ยงดูอบรมสั่งสอน รวมทั้งสนับสนุนและให้กำลังใจแก่ผู้วิจัยเสมอจนสำเร็จการศึกษา

สารบัญ

หน้า

สารบัญ (ต่อ)

บทที่ หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

อินเตอร์เน็ตคือแหล่งที่บรรจุข้อมูลปริมาณมหาศาลไว้ภายใน ซึ่งมีทั้งดีและไม่ดีปะปนกันอยู่และนั่นได้นำไปสู่ปัญหาของการคัดสรรข้อมูลที่เราต้องการจริง ๆ มาใช้งาน การที่มีแหล่งข้อมูลจำนวนมากเกินไปกลายเป็นเรื่องลำบากของผู้ใช้งานที่จะสืบเสาะหาสิ่งหาสิ่งที่อยากจะได้รับจากอินเตอร์เน็ต จึงเกิดมีระบบค้นคืนสารสนเทศบนอินเตอร์เน็ตเพื่อช่วยในการค้นคืนข้อมูลได้สะดวกและรวดเร็วยิ่งขึ้น อย่างไรก็ดีผลลัพธ์ที่ได้จากระบบค้นคืนข้อมูลสารสนเทศในปัจจุบันยังเป็น รายการของเอกสารจำนวนมากและบางครั้งเป็นเอกสารที่ไม่ตรงกับความต้องการจริงของผู้ใช้ด้วย ทำให้ผู้ใช้ต้องมาคัดเลือกหาเอกสารที่ตรงกับความต้องการจริง ๆ ด้วยตัวเอง นับว่ายังเป็น เรื่องที่ยุ่งยากอยู่มากทีเดียว

การค้นคืนสารสนเทศข้ามภาษา (Cross-Language Information Retrieval) หมายถึงการค้นคืนสารสนเทศซึ่งภาษาที่แสดงในเอกสารไม่ตรงกับภาษาที่แสดงในการสอบถาม ปัจจุบันเอกสารทางวิชาการในประเทศไทยมักจะจัดทำทั้งในรูปภาษาไทยและภาษาอังกฤษ เพื่อประโยชน์ในการเผยแพร่ทั้งภายในและภายนอกประเทศ ซึ่งเอกสารเหล่านี้โดยเฉพาะอย่างยิ่งเอกสาร ทางด้านวิทยาศาสตร์และวิศวกรรมศาสตร์โดยมากแล้วมักจะปรากฏคำนามเฉพาะ (Proper Noun) และคำศัพท์เทคนิคต่าง ๆ เป็นจำนวนมาก ซึ่งจะพบได้ทั้งในรูปของคำในภาษาอังกฤษ คำภาษาไทยทับศัพท์อังกฤษ คำภาษาอังกฤษทับศัพท์ภาษาไทย หรือคำในภาษาไทยเอง ดังนั้นถ้าระบบค้นคืนสารสนเทศไม่สนับสนุนการทำงานข้ามภาษาก็จะทำให้ประสิทธิภาพในการ ค้นคืนต่ำและใช้ประโยชน์จากสารสนเทศที่มีอยู่ได้ไม่เต็มที่

จากปัญหาต่าง ๆ ที่กล่าวมาข้างต้น จึงเกิดมีงานวิจัยชิ้นนี้ที่มุ่งเน้นเพื่อพัฒนาระบบค้นคืนสารสนเทศข้ามภาษาไทย-อังกฤษ โดยเน้นการค้นคืนคำทับศัพท์แล้วนำผลลัพธ์ที่ได้จากการ ค้นคืนมาจัดเป็นกลุ่มหรือรวมกลุ่มของเอกสารที่มีความคล้ายคลึงกันเข้าไว้ด้วยกัน เพื่อนำเสนอเป็นผลลัพธ์สุดท้ายต่อผู้ใช้ โดยจะช่วยให้ผู้ใช้สามารถค้นหาข้อมูลที่ต้องการจริง ๆ ได้ง่ายและ รวดเร็วขึ้นด้วยการใช้เทคนิคการจัดกลุ่มข้อมูล (Clustering)

1.2 วัตถุประสงค์ของการวิจัย

เพื่อออกแบบและพัฒนาระบบค้นหา (Search Engine) ข้อมูลสารสนเทศข้ามภาษาไทย-อังกฤษสำหรับคำทับศัพท์และแสดงผลลัพธ์ด้วยการใช้วิธีการจัดกลุ่มข้อมูล (Clustering)

1.3 ขอบเขตของการวิจัย

1. สร้างระบบค้นหาสำหรับคำทับศัพท์ภาษาไทย-อังกฤษเท่านั้นโดยใช้ หลักเกณฑ์การเข้ารหัสคำทับศัพท์ระหว่างคำภาษาอังกฤษกับคำภาษาไทยเท่านั้น
2. คำศัพท์ในภาษาอังกฤษที่ใช้ไม่รวมถึงคำย่อและรัสพจน์ (Acronym)
3. คำทับศัพท์อังกฤษ-ไทยที่ใช้ในการทดสอบขั้นตอนวิธีจะต้องใช้หลักเกณฑ์การทับศัพท์ของราชบัญฑิตยสถาน
4. แสดงผลลัพธ์ด้วยการจัดกลุ่มข้อมูลแบบซัฟฟิกทรี (Suffix Tree Clustering)

1.4 ขั้นตอนและการดำเนินการวิจัย

1. ศึกษาขั้นตอนการทำงานของระบบค้นหา
2. ศึกษาขั้นตอนวิธีการค้นคืนสารสนเทศข้ามภาษา
3. ศึกษาหลักเกณฑ์การทับศัพท์
   3.1 จากภาษาอังกฤษเป็นภาษาไทย
   3.2 จากภาษาไทยเป็นภาษาอังกฤษ
4. ศึกษาภาษาที่ใช้ในการพัฒนาระบบ
5. ศึกษาวิธิการจัดกลุ่มข้อมูล (Clustering) แบบต่าง ๆ
6. ทำการออกแบบและพัฒนาระบบค้นหาสารสนเทศข้ามภาษาไทย-อังกฤษ สำหรับคำทับศัพท์และแสดงผลลัพธ์ด้วยวิธีการจัดกลุ่มข้อมูล (Clustering) ที่ได้เลือกแล้ว
7. ทดสอบและปรับปรุงคุณภาพของโปรแกรม
8. สรุปผลการวิจัยและจัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ระบบค้นคืนสารสนเทศข้ามภาษาไทย-อังกฤษสำหรับคำทับศัพท์และแสดงผลลัพธ์ด้วยวิธีการจัดกลุ่มข้อมูลแบบซัฟฟิกทรี
2. เป็นแนวทางในการพัฒนาระบบค้นหาสารสนเทศข้ามภาษาอื่น ๆ นอกจากภาษาไทย-อังกฤษได้อีก
3. เป็นแนวทางในการพัฒนาระบบค้นหาสารสนเทศให้แสดงผลลัพธ์ด้วยเทคนิควิธีการจัดกลุ่มข้อมูลแบบอื่น ๆ
4. เป็นแนวทางในการพัฒนาระบบค้นหาหรือระบบค้นหาสารสนเทศภาษาไทยได้ต่อไป

บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

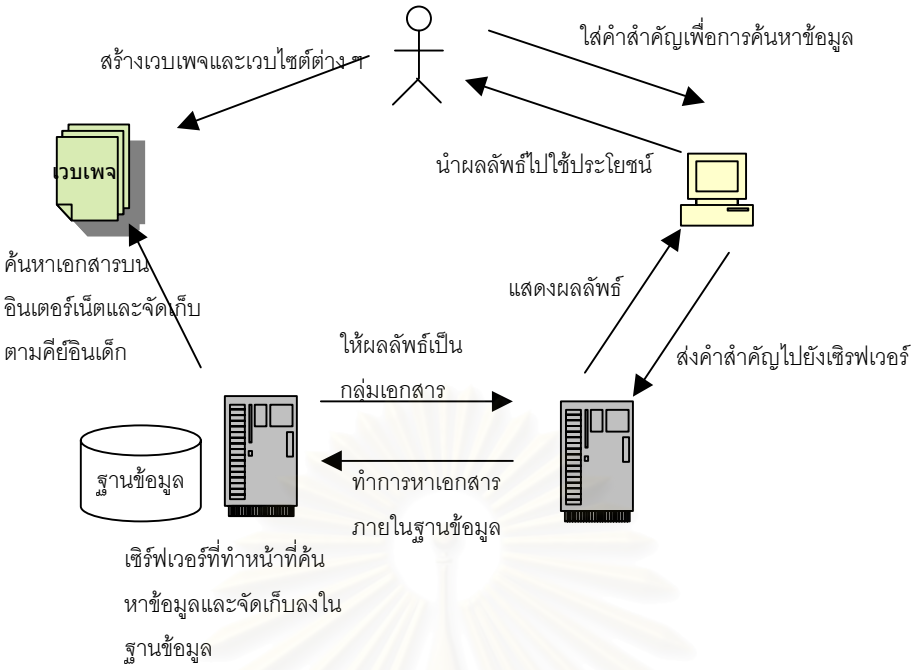
บทนี้กล่าวถึงพื้นฐานความรู้และทฤษฎีที่ใช้ในงานวิจัย เริ่มจากการกล่าวถึงส่วนประกอบและวิธีการทำงานของระบบค้นหาข้อมูล (Search Engine) สรุปลักษณะโดยรวมของการทำงานและเทคนิคในการการค้นคืนข้ามภาษา การตัดคำภาษาไทย การเข้ารหัสคำทับศัพท์ และการจัดกลุ่มผลลัพธ์ที่ได้จาก การค้นหาเพื่อนำมาเสนอเป็นผลลัพธ์ให้ผู้ใช้

2.1 ระบบค้นหาข้อมูล (Search Engine)

ระบบค้นหาข้อมูลหรือที่นิยมเรียกกันว่า เสริชเอนจิน (Search Engine) คือระบบที่ทำงานผ่านโปรโตคอล HTTP เพื่อค้นหาเวบเพจต่างๆให้ผู้ใช้ระบบค้นหาข้อมูล โดยทั่วไปจะมีการเก็บรายชื่อเว็บไซต์และรายละเอียดที่อยู่ภายในเก็บไว้ในฐานข้อมูลขนาดมหึมาของมันเอง เมื่อมีคนเข้ามาเรียกค้นก็จะมีการหาข้อมูลจากฐานข้อมูลมาให้ (ไม่ได้เป็นการเรียกค้นจากอินเตอร์เน็ตในทุกครั้งที่มีคนเรียกใช้)

แต่การที่จะมีรายชื่อเวบไซต์จำนวนมหาศาลได้ไม่ใช่เรื่องง่าย ซึ่งผู้ที่ทำหน้าที่เก็บรายชื่อเวบไซต์ให้กับระบบค้นหาข้อมูลคือโปรแกรมขนาดเล็ก ๆ ที่เรียกว่า สไปเดอร์ (spider) หรือ โรบอท (robot) หรือ คลอว์เลอร์ (crawler) ซึ่งผู้สร้างระบบค้นหาได้ปล่อยออกไปในอินเตอร์เน็ต ให้มันท่องเที่ยวไปในเวบไซต์ทีละเวบไซต์คืบคลานไปเรื่อยๆ ขณะเดียวกันมันก็จะส่งข้อมูล เกี่ยวกับเวบไซต์ที่มันเข้าไปเยือนกลับมาที่ระบบค้นหาข้อมูล เมื่อได้ข้อมูลจากอินเตอร์เน็ตก็จะ มีการจัดเก็บข้อมูลเหล่านั้นลงในฐานข้อมูลรอให้ผู้ใช้บริการมาเรียกค้น แต่ปัจจุบันจำนวนเวบไซต์ มีมากมายเพิ่มขึ้นทุกวันกว่าที่สไปเดอร์ (spider) จะเยี่ยมเยียนครบก็ใช้เวลานานจึงมีโอกาส ที่บางเวบไซต์มีการเปลี่ยนแปลงไป

รูปที่ 2.1 แสดงการทำงานของระบบค้นหาข้อมูลและการค้นหาข้อมูลโดยผู้ใช้เอง โดยจุดเริ่มต้นก็คือมีคนสร้างเวบเพจและเวบไซต์ต่าง ๆ บนอินเตอร์เน็ตตลอดเวลา เมื่อถึงเวลาที่ ระบบค้นหาต้องส่งสไปเดอร์ (spider) ออกไปเก็บข้อมูลของเวบเพจมาอัพเดตและเพิ่มเติมลงใน ฐานข้อมูลของมันเอง ในขณะที่คนที่ต้องการค้นหาข้อมูลจะมาทำการค้นในระบบค้นหาซึ่งระบบ ค้นหาจะนำคำสำคัญของผู้ใช้ส่งไปหาเอกสารที่ตรงกับความต้องการในฐานข้อมูลของมันเองและ ส่งผลลัพท์ที่ได้กลับไปหาผู้ใช้



รูปที่ 2.1 การทำงานของระบบค้นหาข้อมูล

2.1.1 การค้นหาข้อมูลบนอินเตอร์เน็ตเพื่อเก็บรายละเอียดลงในฐานข้อมูลของระบบ ค้นหา (Crawling the web)

ทำอย่างไรระบบค้นหาจึงสามารถพบข้อมูลบนอินเตอร์เน็ตปริมาณมหาศาลที่มีทั้งการเพิ่มข้อมูลใหม่ๆ เข้าไปทุกวันและมีการปรับปรุงเปลี่ยนแปลงอยู่ตลอดเวลาแล้วเก็บข้อมูลเหล่านั้นลงในฐานข้อมูลของตัวเอง

ระบบค้นหาจะสั่งให้โปรแกรมที่เรียกว่าสไปเดอร์(spider) ผ่านไปยังเซิรฟเวอร์ใดเซิรฟเวอร์หนึ่งในการเริ่มต้นเก็บข้อมูล จากนั้นสไปเดอร์จะท่องไปยังไฮเปอร์ลิงค์ภายในเวบไซต์ไปเรื่อยๆ และเก็บข้อมูลของเอกสารต่างๆ มาทำการจัดทำดัชนีก่อนเก็บลงในฐานข้อมูล อัลกอริทึมง่ายๆของขั้นตอนนี้สามารถแสดงได้ดังรูปที่ 2.2 ด้านล่างนี้

1. ส่ง โปรแกรม SPIDER ไปเริ่มต้นค้นที่เซิรฟเวอร์
2. ถ้าในฐานข้อมูล URLของเซิรฟเวอร์ (URL Pool) ยังมีข้อมูลของเอกสารและ URL อยู่
   2.1 เก็บชื่อ URL ไว้
   2.2 ดาวน์โหลดเอกสารที่เกี่ยวกับ URL นั้น
   2.3 เก็บข้อมูลลิงค์ต่าง ๆ ที่มีภายใน URL นั้น
   2.4 จัดทำดัชนีของ URL กับข้อมูลของ URL ลงในฐานข้อมูลของระบบค้นคืน

2.5 พิจารณาลิงค์ต่าง ๆ ที่ได้จาก 2.3 ทีละลิงค์จนหมด

2.5.1 ตรวจสอบว่า ถ้าลิงค์นั้นยังไม่ถูกจัดเก็บลงในฐานข้อมูลของระบบค้นคืน

2.5.2 เพิ่มลิงค์นั้นเข้าไปในฐานข้อมูล URLของเซิรฟเวอร์ (URL Pool)

2.5.3 กลับไปพิจารณาลิงค์ที่เหลือในข้อ 2.5.1

3. กลับไปพิจารณา URL ที่เหลือในข้อ 2

4. จบการทำงาน

รูปที่ 2.2 อัลกอริทึมการค้นหาข้อมูลในแต่ละเซิรฟเวอร์

2.1.2 **การเก็บข้อมูลลงในฐานข้อมูลของระบบค้นหา** (Storing the information)

ปัจจุบันอินเตอร์เน็ตมีข้อมูลจำนวนมากปรากฏอยู่ทำให้ระบบค้นหาต้องเก็บข้อมูลจำนวนมากด้วย จะเห็นได้ว่าระบบค้นหาต้องใช้เนื้อที่ของฐานข้อมูลมากพอสมควรดังนั้นจึงควร มีหลักการ ในการจัดเก็บข้อมูลให้มีประสิทธิภาพที่ดีที่เหมาะสมกับเนื้อที่และง่ายต่อการค้นหา เช่น ในงานวิจัยชิ้นนี้จะจัดเก็บข้อมูลเป็น 2 แบบคือ เก็บเป็นฐานข้อมูลของคำที่มีลิงค์เชื่อม ต่อไปยังเอกสารจริงที่มีคำนั้นอยู่ซึ่งถูกเก็บอยู่ในฐานข้อมูลของเอกสารอีกที่หนึ่งเป็นต้น การจัดเก็บ แบบนี้ช่วยในการค้นหาได้รวดเร็ว เนื่องจากเมื่อผู้ใช้ค้นหาจากข้อคำถามระบบจะมาตรวจสอบใน ฐานข้อมูลคำก่อนแล้วจึงไปค้นจากฐานข้อมูลที่เก็บเอกสารจริงต่อไป

2.1.3 **การค้นหาข้อมูลในฐานข้อมูลของระบบค้นหา** (Querying the web)

วิธีที่ง่ายที่สุดในการค้นหาเอกสารของระบบค้นหาส่วนใหญ่คือ ระบบค้นหาจะรับ ข้อคำถาม (keyword) นำไปตัดแบ่งเป็นคำ จากนั้นจะนำคำที่ได้ไปเปรียบเทียบกับคำในเอกสาร ทีละคำหากพบตรงกันจะแสดงเอกสารนั้นเป็นผลลัพธ์แก่ผู้ใช้ วิธีการนี้ไม่ได้คำนึงถึงคุณภาพ ของผลลัพธ์ที่ได้และบางครั้งผลลัพธ์ที่ได้ก็แทบไม่ตรงกับความต้องการจริง ๆ ของผู้ใช้เลย ปัจจุบันจึงมีการประยุกต์ใช้อัลกอริทึมหลายแบบ เพื่อมาช่วยให้กระบวนการค้นหาข้อมูลเป็นไป อย่างมีประสิทธิผลมากขึ้น กล่าวคือได้ข้อมูลที่ตรงกับข้อคำถามและความต้องการของผู้ใช้มาก ที่สุดเช่น ใช้หลักการแบบเวคเตอร์สเปซ (vector space) โดยระบบค้นหาจะให้น้ำหนัก ของคำแต่ละคำในเอกสารแล้วมาประมวลผลคำเพื่อเรียงลำดับตามความสำคัญของคำในเอกสาร ว่าคำใดตรงกับความต้องการของผู้ใช้มากที่สุดและรองลงมา เป็นต้น

2.1.4 **การแสดงผลลัพธ์ของการสืบค้น** (Presentation of query result)

ระบบค้นหาสามารถค้นคืนเอกสารให้กับผู้ใช้จำนวนมากและผลลัพธ์ที่ได้ส่วนใหญ่มักไม่ตรงกับความต้องการของผู้ใช้ ผู้ใช้ที่มีประสบการณ์ในการสืบค้นจะสามารถสืบค้นข้อมูลด้วยการระบุข้อคำถามที่ชัดเจนและค่อนข้างครอบคลุมต่อความต้องการพอสมควรเช่น ถ้าต้องการค้นหาเกี่ยวกับการจัดกลุ่มข้อมูลแบบซัฟฟิกทรี (Suffix Tree) ผู้ใช้ก็ต้องระบุคำว่า “Sufix Tree” and “Clustering” ในส่วนรับข้อคำถาม เป็นต้น แต่สำหรับผู้ใช้ที่ไม่สามารถระบุข้อคำถามได้ครอบคลุมกับความต้องการได้ คงต้องเผชิญกับปัญหาจากการค้นหาเอกสารจำนวนมหาศาลเช่น ระบุคำว่า “Suffix Tree” เท่านั้น ระบบค้นคืนอาจให้ผลลัพท์กลับมาเป็นเรื่องที่เกี่ยวกับเรื่องอื่น ๆ ที่เกี่ยว กับซัฟฟิกทรี (Siffix Tree) แต่ไม่ใช่เรื่องการจัดกลุ่มข้อมูลมาด้วย เป็นต้น การแสดงผลลัพธ์จากการสืบค้นของระบบค้นหามีหลายแบบ ดังต่อไปนี้

1) **การแสดงผลลัพธ์แบบ** Ranked list

เป็นเทคนิคที่ระบบค้นหาทั่วไปเช่น Yahoo [1] , Altavista [2] , Google[3] , Excite [4] นิยมใช้ในการแสดงผลลัพธ์ กลุ่มของเอกสารที่ตรงกับข้อคำถามของผู้ใช้จะถูกเรียงลำดับจาก มากไปน้อยขึ้นอยู่กับเอกสารนั้นมีคำสำคัญ (keyword) ที่ผู้ใช้ต้องการค้นหามากน้อยแค่ไหน ผลลัพธ์แต่ละตัวจะประกอบไปด้วยหัวข้อ (title) , URL และข้อความสั้นๆ ซึ่งอธิบายหรือแสดง ลักษณะของเอกสารนั้นๆ (snippet) การแสดงผลลัพธ์ด้วยวิธีนี้มีข้อเสียคือ ผู้ใช้ต้องเลือกหา ข้อมูลที่ต้องการจากรายการเอกสารจำนวนมาก ซึ่งบางครั้งที่ความสัมพันธ์ระหว่างเอกสารกับ ข้อคำถามไม่ชัดเจนและการเรียงลำดับผลลัพธ์ที่อาศัยการวัดความสำคัญจากจำนวนคำสำคัญที่ ปรากฏในแต่ละเอกสารเท่านั้นซึ่งไม่ได้คำนึงถึงความต้องการจริงของผู้ใช้เลย

รูปที่ 2.3 ตัวอย่างการแสดงผลลัพธ์แบบ Ranked list

2) **การแสดงผลลัพธ์แบบไดเรคทอรีเวบ**(Web Directories)

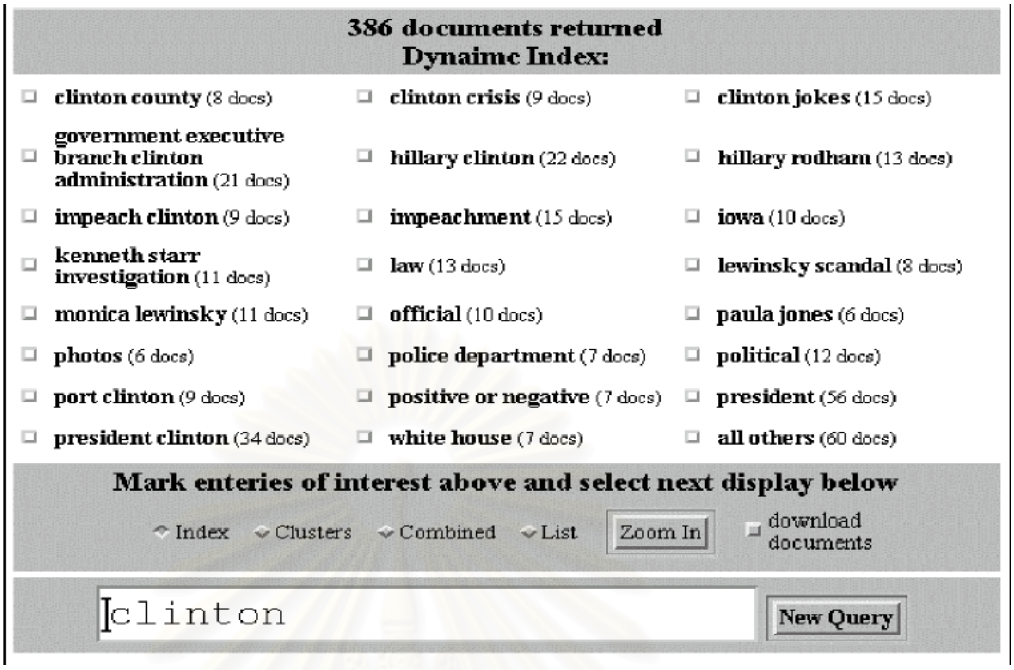
Yahoo ถือได้ว่าเป็นเวบไซต์แรกที่ใช้เทคนิคการแสดงผลลัพธ์แบบไดเรคทอรีเวบ โดยการนำเอกสารต่าง ๆ มาจัดกลุ่มแบ่งเป็นประเภทดังแสดงในรูปที่ 2.4 ผู้ดูแลระบบค้นหาจะพิจารณาเอกสารที่มีว่าอยู่ในประเภทใดแล้วจะจัดให้เอกสารถูกแสดงภายในกลุ่มนั้น วิธีนี้คล้ายกับ การแสดงผลลัพธ์แบบ Ranked list แต่ต่างกันตรงที่วิธีนี้การจัดกลุ่มหรือประเภทของเอกสาร ทำโดยวิจารญาณของมนุษย์ แน่นอนว่าการจัดประเภทผลลัพธ์แบบนี้ต้องเสียเวลามาก

รูปที่ 2.4 ตัวอย่างการแสดงผลลัพธ์แบบไดเรคทอรีเวบ (Web directories)

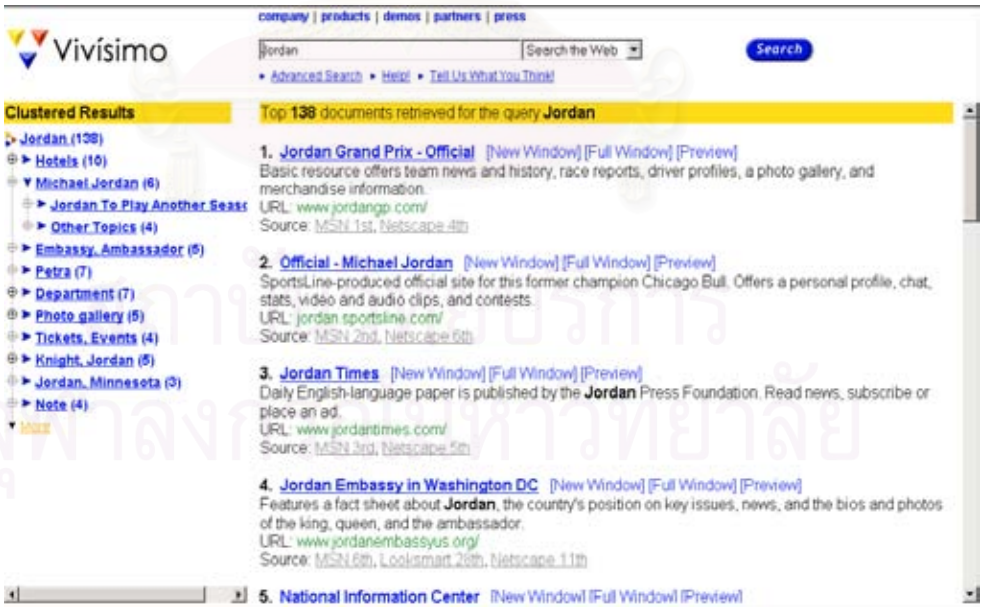
3) **การแสดงผลลัพธ์แบบการจัดกลุ่มข้อมูลแบบคลัสเตอร์ลิ่ง** (Clustering)

เป็นการปรับปรุงลักษณะการแสดงผลลัพธ์แบบ Ranked list โดยสร้างลำดับชั้นของการแสดงผลลัพธ์ วิธีนี้ช่วยให้ผู้ใช้สามารถค้นหาข้อมูลได้สะดวกและรวดเร็วยิ่งขึ้น โดยการที่ผู้ใช้เริ่มมองหาประเภทของเอกสารที่ต้องการก่อน แล้วค่อย ๆ ค้นหาไปในระดับที่เฉพาะเจาะจงลงไปอีก ตามลำดับชั้นภายใต้ประเภทเอกสารนั้น ๆ จะเห็นว่าผู้ใช้สามารถตัดประเภทเอกสารที่ไม่ตรง ตามความต้องการออกไปได้เลยในขณะทำการค้นหา อัลกอริทึมที่ใช้ในการแสดงผลลัพธ์ แบบนี้ เช่น Hierarchical Agglomerative Clustering (HAC) [5] ,K-means [5] เป็นต้น เทคนิคนี้ช่วยสนับสนุนหลักการของการจัดกลุ่มผลลัพธ์หลังการค้นคืน (post-retrieval document clustering system) ดังนี้

- การหาความสัมพันธ์ระหว่างเอกสารจากผลลัพธ์ที่ได้มาจากการค้นคืน (Relevance)
  วิธีนี้จะจัดกลุ่มเอกสารที่มีความหมายในเรื่องเดียวกันหรือใกล้เคียงให้อยู่ในกลุ่มเดียวกัน และ อยู่ต่างกลุ่มกันกับเอกสารที่ไม่เกี่ยวข้องกัน
- มีรูปแบบการแสดงผลลัพธ์อย่างมีประสิทธิภาพ (Browsable Summaries)
  การแสดงผลลัพธ์ด้วยการจัดกลุ่มข้อมูลภายใต้หัวข้อที่สร้างจากคำภายในเอกสารกลุ่มนั้น ๆ อย่างรัดกุม มีความหมายชัดเจนเพื่อให้ผู้ใช้สามารถเลือกหาเอกสารได้ง่ายและรวดเร็วยิ่งขึ้นได้
- ทำงานกับคำอธิบายเอกสารอย่างย่อ (Snippet tolerance)
  การจัดกลุ่มและสร้างหัวข้อกลุ่มข้อมูลจากคำอธิบายย่อของแต่ละเอกสาร ซึ่งให้ ความหมายที่ถูกต้องและช่วยลดเวลาการทำงาน
- ใช้เวลาน้อย (Incrementality)
  ขณะทำงานกับคำอธิบายย่อของเอกสารหนึ่ง ๆ เพื่อสร้างผลลัพธ์ที่จะแสดงให้ผู้ใช้ ระบบก็จะดาวน์โหลดเอกสารอื่นได้ด้วย โดยไม่ต้องรอให้ระบบค้นหาเอกสารทั้งหมด จนจบก่อน วิธีการนี้ช่วยให้ระบบประหยัดเวลาในการค้นหาและช่วยให้จัดรูปแบบ แสดงผลลัพธ์ได้เร็วและมีประสิทธิภาพยิ่งขึ้น
- การยอมให้เอกสารปรากฏอยู่ในหัวข้อกลุ่มข้อมูลมากกว่า 1 กลุ่ม (Overlapping clustering)
  นอกจากระบบสามารถจัดให้เอกสารที่มีความหมายเดียวกันหรือใกล้เคียงกันอยู่ในกลุ่มเดียวกันได้แล้ว ในขณะเดียวกันระบบควรยินยอมให้เอกสารหนึ่ง ๆ สามารถถูกแสดง อยู่ในกลุ่มอื่นได้ด้วย ถ้าเอกสารนั้นมีความหมายเดียวกันหรือใกล้เคียงกับกลุ่มเอกสารอื่น ๆ เพราะในความ เป็นจริงเอกสารฉบับหนึ่งสามารถมีได้หลายความหมายและหลายหัวข้อ
- แสดงคำอธิบายกลุ่มข้อมูลให้เข้าใจง่าย (Simple cluster definitions)
  คำอธิบายกลุ่มข้อมูลที่เข้าใจง่ายช่วยให้ผู้ใช้ค้นหาข้อมูลที่สนใจได้ดีและมั่นใจได้มากขึ้น เนื่องจากวิธีนี้ใช้คำจากเอกสารต่างมาแสดงเป็นหัวข้อให้ผู้ใช้เลือก
- การใช้กลุ่มคำ (Use of phrases)
  การพิจารณาถึงกลุ่มคำคือคำนึงถึงความสัมพันธ์ของคำภายในเอกสาร แทนที่จะพิจารณาแต่คำแต่ละคำแยกจากกันไปเลย ช่วยให้การจัดกลุ่มข้อมูลและแสดงคำอธิบายกลุ่มข้อมูลของระบบมีประสิทธิภาพที่ดียิ่งขึ้น ปัจจุบันมีการนำเอาเทคนิคการจัดกลุ่มข้อมูล แบบคลัสเตอร์ลิ่งเข้ามาจัดการผลลัพธ์ภายในระบบค้นหาต่าง ๆ เช่น ในระบบค้นหา Grouper [6] ซึ่งเป็นงานวิจัยของ Zamir และ Etzioni [7] ดังแสดงในรูปที่ 2.5 และ Vivismo [8] ดังแสดงในรูปที่ 2.6



รูปที่ 2.5 รูปแบบการจัดกลุ่มข้อมูลของระบบค้นหา Grouper



รูปที่ 2.6 รูปแบบการจัดกลุ่มข้อมูลของระบบค้นหา Vivisimo

2.2 การค้นคืนข้ามภาษา

การค้นคืนสารสนเทศข้ามภาษา [9] หมายถึง การค้นคืนสารสนเทศที่อนุญาตให้ผู้ใช้สามารถระบุข้อคำถามด้วยภาษาใดภาษาหนึ่ง แล้วระบบจะทำการคืนสารสนเทศข้ามภาษา ให้โดยอัตโนมัติ โดยทั่วไปแล้ว สามารถแบ่งการค้นคืนสารสนเทศข้ามภาษาออกเป็น 2 วิธีคือ

1. การแปลข้อคำถาม คือ การแปลภาษาของข้อคำถามที่ผู้ใช้ระบุขณะที่ทำการสืบค้น ให้เป็นภาษา เดียวกับที่จัดเก็บในเอกสารก่อน แล้วจึงทำการสืบค้น
2. การแปลเอกสาร คือ การแปลภาษาของเอกสารทั้งหมดให้เป็นภาษาเดียวกับข้อคำถามก่อนแล้วจึงทำการสร้างดัชนี

งานวิจัยทางด้านการค้นคืนสารสนเทศข้ามภาษาที่ผ่านมานั้น สามารถแบ่งออกเป็น 3 แนวคิดหลัก ๆ คือ

1. การแปลข้อความโดยใช้เทคนิคการแปลภาษาด้วยเครื่อง (MachineTranslation) คือ การแปลภาษาที่เอกสารหรือข้อคำถามด้วยเครื่องแปลภาษา ซึ่งข้อด้อยของวิธีการนี้ ขึ้นอยู่กับคุณภาพ ของการแปลภาษา
2. การใช้ฐานพจนานุกรมหลายภาษา (MultilingualDictionary) คือ การใช้พจนานุกรมหลายภาษา ในลักษณะของบัญชีคำอรรถาภิธานหรือการสร้างบัญชีคำพ้องความหมาย แต่เป็นการพ้องความหมายข้ามภาษา เช่น คำว่า “Food” จะพ้องความหมายกับคำว่า “อาหาร” เป็นต้น
3. การใช้ฐานคำแบบขนาน (Parallel Corpus-Based) เป็นอีกทางเลือกของการใช้บัญชีคำ-อรรถาภิธาน คือการใช้ประโยชน์จากฐานข้อมูลที่มีอยู่แล้ว ซึ่งจัดเก็บในลักษณะสองภาษาควบคู่กันไปตัวอย่างเช่น ข้อมูลหนังสือที่มีการจัดเก็บชื่อเรื่องที่เป็นภาษาไทยและภาษาอังกฤษควบคู่กันไป เป็นต้น โดยจะนำข้อมูลแต่ละคู่มาจัดเก็บเป็นฐานคำ แบบขนาน

การถอดอักษร (Transliteration)

การถอดอักษร [9] หมายถึง การนำคำในภาษาหนึ่งมาเขียนด้วยตัวอักษรอีกภาษาหนึ่ง แบบอักษรต่ออักษร โดยพยายามใช้หน่วยเสียงของอักษรทั้งสองภาษาใกล้เคียงกันมากที่สุด ในการถอดอักษรแบ่ง เป็น 3 ขั้นตอนหลัก ๆ ดังนี้

1. ถ่ายเสียงจากหน่วยอักษรในภาษาต้นแบบ (Source Language) ไปเป็นหน่วยเสียง ในภาษาต้นแบบ
2. เทียบหน่วยเสียงในภาษาต้นแบบกับหน่วยเสียงในภาษาเป้าหมาย (Target Language) โดยพยายามใช้หน่วยเสียงที่ใกล้เคียงกันมากที่สุด
3. ถอดหน่วยเสียงในภาษาเป้าหมายเป็นหน่วยอักษรในภาษาเป้าหมาย

ปัญหาต่าง ๆ ในการถอดอักษร

- ระบบอักษรของภาษาต้นแบบมีความสัมพันธ์ของหน่วยอักษรกับหน่วยเสียงเป็นแบบหนึ่งตัวอักษร แทนหลายหน่วยเสียง เช่น ในภาษาอังกฤษ “C” แทนด้วย /k/ หรือ /s/ และหนึ่งหน่วยเสียงแทน หลายตัวอักษร เช่น ในภาษาอังกฤษ “N,TN,GN,PN” แทนด้วย /n/
- ระบบอักษรของภาษาเป้าหมายมีความสัมพันธ์ของหน่วยอักษรกับหน่วยเสียงหลายแบบเหมือนในภาษาต้นแบบ เช่น ในภาษาไทย “ร, ฤ, หร” แทนด้วย /r/ และ “ฉ, ช, ฌ” แทนด้วย /ch/ เป็นต้น
- การแบ่งพยางค์ในภาษาต้นแบบเมื่อมีพยัญชนะตัวเดียวอยู่ระหว่างสระ เช่น คำว่า money ในภาษาอังกฤษ จะแบ่งพยางค์อย่างไร จะถอดพยัญชนะซ้ำสองตัวเพื่อให้อ่านได้สะดวกเป็น มัน-นีย์ หรือจะถอดอักษรเพียงตัวเดียวตามที่ปรากฏในภาษาอังกฤษเป็น มะ-นีย์ หรือ มัน-อีย์
- ปัญหาอันเนื่องมาจากช่วงเวลาของการยืมคำทับศัพท์ คำทับศัพท์บางคำยืมมาเป็นเวลานาน ซึ่งในอดีตมีหลักเกณฑ์การทับศัพท์ไม่ตรงกับหลักเกณฑ์ในปัจจุบัน เช่น “C” ที่แทน /k/ ในอดีตนิยมถอดเป็นอักษร “ก” เช่น กุ๊ก (Cook) กัปตัน(Captain) กะรัต(Carat) แก๊ป(Cap) เป็นต้น แต่ปัจจุบัน “C” ที่แทน /k/ มักจะถอดเป็น “ค” ในตำแหน่งพยัญชนะต้น เช่น คอนโดยมิเนียม (Condominium) แคปซูล(Capsule) แคร็อต(Carrot) เป็นต้น

การใช้ตัวอักษรโรมันเพื่อการถ่ายเสียง (Romanization)

การใช้ตัวอักษรโรมันเพื่อการถ่ายเสียง คือ การถ่ายเสียงตัวอักษรของภาษาอื่น ๆ ที่ไม่ใช่อักษรโรมัน เช่น ไทย จีน ญี่ปุ่น ฯลฯ ให้เป็นตัวอักษรโรมัน เพื่อให้ผู้ที่ไม่รู้จักภาษานั้น ๆ สามารถอ่านออกเสียงได้ เช่น คำว่า “อยุธยา” สามารถถ่ายเสียงเป็น “AYUTTAYA” เป็นต้น การเขียนชื่อเฉพาะต่าง ๆ จากภาษาไทยด้วยอักษรโรมันนั้นมีปัญหามาก เช่น คำว่า “ไทยทนุ” เขียนเป็น “Thai Danu” ซึ่งการใช้ตัวอักษร “D” แทนตัว “ท” นั้นเป็นการเขียนโดยอาศัย หลักเกณฑ์ของภาษาบาลี

จากปัญหาต่าง ๆ ที่เกิดขึ้น ทางราชบัณฑิตยสถานจึงได้พยายามสร้างระบบมาตรฐาน ในการใช้ตัวอักษรโรมันเพื่อการถ่ายเสียงสำหรับตัวอักษรไทย โดยมีแนวคิดว่าควรยึดหลัก ภาษาไทยเป็นที่ตั้งและหาวิธีการถอดอักษรตามวิธีเขียนและออกสำเนียงในภาษาไทย เพราะถ้ายึด หลักการถอดอักษรตามศัพท์ในภาษาเดิมซึ่งเป็นที่มาของคำไทยแล้ว อักษรไทยตัวเดียวกันอาจ ถอดเป็นอักษรโรมันต่างกันก็ได้เช่น ทร จะถอดอักษรเป็น Thon , S, Tr., Tara แต่ถ้าจะถอดอักษรตามเสียงอย่างเดียวเท่านั้น เมื่อเขียนกลับเป็นภาษาไทยก็เขียนได้หลายอย่าง เช่น Ban Ma เขียนเป็น บ้านมา , บ้านม่า หรือ บ้านหมาได้ เป็นต้น

ในที่สุด ทางราชบัณฑิตยสถานจึงได้กำหนดระบบการใช้ตัวอักษรโรมันเพื่อการถ่ายเสียงสำหรับตัวอักษรไทยออกเป็น 2 ระบบ คือ ระบบทั่วไป (General System) และระบบเฉพาะ (Precise System) โดยระบบทั่วไปจะใช้สำหรับกรณีที่การออกเสียงสำคัญกว่าการเขียนตัวสะกด ซึ่งจะอาศัยหลักการออกเสียงเป็นสำคัญ ต้องสอดคล้องกับไวยากรณ์ของไทยและสามารถขยายเป็นระบบเฉพาะได้ เช่น คำว่า “กษัตริย์” ถอดเป็น Kasat ส่วนระบบเฉพาะจะใช้ในกรณี ที่จะแสดงตัวอักษรให้ละเอียดแม่นยำเพื่อให้คงความหมายของคำนั้นไว้เช่น คำว่า “กษัตริย์” ถอดเป็น Kasatriy

หลักเกณฑ์การทับศัพท์

ในปัจจุบันได้มีการบัญญัติศัพท์วิชาการขึ้นใช้กันอย่างแพร่หลายและในการบัญญัติศัพท์นั้นบางครั้งไม่สามารถหาคำไทยมาใช้ได้ตรงความหมายที่ต้องการ จึงต้องใช้วิธีทับศัพท์ นอกจากนี้การเขียนคำวิสามานยนามต่างๆ เช่น ชื่อคน ชื่อสถานที่ ก็ต้องใช้วิธีทับศัพท์ เช่นเดียวกันเพื่อให้เป็นมาตรฐานเดียวกันในการทับศัพท์ ทางราชบัณฑิตสภานจึงได้กำหนด หลักเกณฑ์การทับศัพท์ไว้ดังนี้

1. การทับศัพท์ให้ถอดอักษรในภาษาเดิมพอควรแก่การแสดงที่มาของรูปศัพท์และให้เขียนในรูปที่อ่านได้สะดวกในภาษาไทย
2. การวางหลักเกณฑ์ได้แยกกำหนดหลักเกณฑ์การทับศัพท์ภาษาต่าง ๆ แต่ละภาษาไป
3. คำทับศัพท์ที่ใช้กันมานานจนถือเป็นคำไทย และปรากฏในพจนานุกรมฉบับราชบัญฑิตยสถานแล้วให้ใช้ต่อไปตามเดิม เช่น ช็อกโกเลต ช็อกโกแลต เชิ้ต แก๊ส ก๊าซ
4. ศัพท์วิชาการซึ่งได้ใช้เฉพาะกลุ่ม ไม่ใช่ศัพท์ทั่วไป อาจเพิ่มหลักเกณฑ์ขึ้นตามความจำเป็น

ขั้นตอนวิธีซาวด์เด็กซ์ภาษาอังกฤษของ Odell และ Russell

M.K. Odell และ R. C. Russell [9] ได้ออกแบบขั้นตอนวิธีการเข้ารหัสชื่อ โดยยึดหลักของการอ่านออกเสียง ทำให้ชื่อที่อ่านออกเสียงเหมือนกันได้รหัสเหมือนกัน หรือที่เรียกว่า “ซาวด์เด็กซ์” (Soundex) ขั้นตอนวิธีดังกล่าวได้ใช้แนวคิดทางภาษาศาสตร์และชวเลขที่ว่า ชื่อในภาษาอังกฤษสามารถจำแนกความแตกต่างได้โดยพิจารณาเพียงพยัญชนะเท่านั้น วิธีการเข้ารหัสจะเริ่มจากการแปลงแต่ละตัวอักษร (ยกเว้นตัวอักษรแรก) ไปเป็นรหัสตัวเลขโดยใช้ตารางการกำหนด รหัสซาวด์เด็กซ์ (ตารางที่ 2.1) จากนั้นจะตัดรหัสตัวเลขศูนย์ออกไป และถ้ารหัสตัวเลขที่อยู่ ตำแหน่งติดกันมีค่าเท่ากันจะเก็บเพียงหนึ่งรหัสเท่านั้น สุดท้ายรหัสซาวด์เด็กซ์ที่ได้คือตัวอักษร ตัวแรกของชื่อตามด้วยรหัสตัวเลขสามตัวแรกที่ได้จากการแปลง ถ้ารหัสยาวไม่พอเติมตัวเลขศูนย์ จนครบสามตัว ตัวอย่างเช่น ALEXANDER แปลงเป็น A425

ตารางที่ 2.1 การกำหนดรหัสซาวด์เด็กซ์

ตัวอักษร	รหัสตัวเลข
A E I O U H W Y	0
B F P V	1
C G J K Q S X Z	2
D T	3
L	4
M N	5
R	6

ขั้นตอนวิธีซาวด์เด็กซ์ดังกล่าวเป็นขั้นตอนวิธีที่ง่ายและทำงานได้รวดเร็ว แต่บางครั้งก็จะพบความผิดพลาดที่เกิดขึ้นเช่น มีชื่อที่ได้รหัสซาวด์เด็กซ์ตรงกันแต่อ่านออกเสียงไม่เหมือนกัน

ขั้นตอนวิธีซาวด์เด็กซ์ภาษาไทย

ปัจจุบันมีงานวิจัยทางด้านซาวด์เด็กซ์ภาษาไทยเพื่อแก้ปัญหาต่าง ๆ เช่น การสืบค้นชื่อและนามสกุลที่อ่านออกเสียงเหมือนกันในทะเบียนรายชื่อ การสืบค้นคำไทยที่มักสะกดผิด การตรวจสอบตัวสะกด เป็นต้น การออกแบบและพัฒนาขั้นตอนวิธีเพื่อแก้ไขปัญหาดังกล่าว จึงแตกต่างกันไป การสืบค้นชื่อและนามสกุลมีการออกแบบขั้นตอนวิธีการเข้ารหัสซาวด์เด็กซ์ ภาษาไทยเพื่อให้ชื่อที่อ่านออกเสียงเหมือนกันได้รหัสตรงกัน โดยใช้หลักเกณฑ์และข้อกำหนด ดัดแปลงมาจากซาวด์เด็กซ์ในภาษาอังกฤษและเพิ่มเติมบางส่วนเพื่อเหมาะสมกับภาษาไทย เช่น การจัดกลุ่มพยัญชนะต้นออกเป็น 20 กลุ่ม (ตามหน่วยเสียง) และพยัญชนะตามออกเป็น 6 กลุ่ม (ตามตัวสะกดแม่ต่าง ๆ) โดยปกติไม่พิจารณา สระ และเสียงวรรณยุกต์ ยกเว้นสระบางตัว ไม่พิจารณาพยัญชนะ ที่มีตัวการันต์กำกับ เป็นต้น

การสืบค้นคำไทยที่มักสะกดผิด มีการออกแบบขั้นตอนวิธีการเข้ารหัสซาวด์เด็กซ์ ภาษาไทยเพื่อให้รหัสที่ได้ตรงกันระหว่างคำที่สะกดถูกและสะกดผิด โดยมีแนวคิดว่าใช้อักขระ ไทยในการเข้ารหัสไม่จำกัดความยาวของรหัส พยายามพิจารณาทุกอักขระและได้เพิ่มกฎเกณฑ์ ต่าง ๆ เพื่อแก้ปัญหาการสะกดผิดโดยเฉพาะ เช่น การเปลี่ยนตำแหน่งสระหน้า (เ- แ- โ- ไ- ใ-) ไปไว้หลังสุดของคำและมีการพิจารณาถึงอักษรควบกล้ำ

2.3 การตัดคำภาษาไทย

เนื่องจากภาษาไทยไม่เหมือนภาษาอังกฤษที่มีการเว้นวรรคระหว่างคำแต่ภาษาไทยจะเขียนเรียงต่อกันไปในประโยคหนึ่ง ๆ อีกทั้งยังประกอบด้วยสระทั้งด้านบนและด้านล่างอีกด้วย ดังนั้นการสร้างระบบค้นหาคำภาษาไทย ต้องมีส่วนที่ทำการตัดคำภายในประโยคออกเป็น คำย่อยเพื่อสามารถเก็บลงในฐานข้อมูลของระบบค้นหาให้สามารถทำการสืบค้นต่อไปได้ ปัจจุบันยังไม่มีอัลกอริทึมที่สามารถตัดคำไทยได้ถูกต้องสมบูรณ์ แต่เวบไซต์ที่สามารถค้นหา คำไทยได้ก็เลือกใช้อัลกอริทึมการตัดคำที่แตกต่างกันไปเช่น อัลกอริทึมการตัดคำมากสุด (Maximal Matching) หรือ อัลกอริทึมการตัดคำยาวที่สุด (Longest Matching)

หลักการตัดคำในภาษาไทยสามารถแบ่งออกได้เป็น 2 หลักการใหญ่คือ หลักการ ตัดคำโดยใช้กฎเกณฑ์ ซึ่งขั้นตอนการตัดคำในยุคแรก ๆ จะใช้วิธีการตรวจสอบกฎเกณฑ์ของคำ ภาษาไทย เช่น กฎเกณฑ์ของตัวอักษรที่อยู่ติดกันหรือกฎเกณฑ์ที่กำหนดโดยราชบัณฑิตยสถาน วิธีการนี้มีข้อจำกัดมากนั่นคือผลของการตัดคำอาจได้เป็นกลุ่มของคำ ซึ่งในความเป็นจริงยัง สามารถตัดคำแยกย่อยออกไปได้อีกนั่นคือความถูกต้องของคำหลังการตัดต่ำ หลักการตัดคำโดย ใช้พจนานุกรมในยุคต่อมา ขั้นตอนวิธีการตัดคำภาษาไทยโดยส่วนใหญ่ซึ่งจะใช้พจนานุกรม เข้าช่วย วิธีการนี้ถึงแม้จะใช้เนื้อที่ความจำหลักมากแต่เป็นวิธีที่ให้ความถูกต้องในการตัดคำสูง วิธีหนึ่ง

ในงานวิจัยนี้ได้เลือกขั้นตอนการตัดคำโดยใช้พจนานุกรม เนื่องจากงานวิจัยที่ผ่านมา ได้แสดงให้เห็นว่าการตัดคำโดยใช้พจนานุกรมสามารถตัดคำได้ถูกต้องมากกว่า และได้ใช้วิธีตัดคำยาวสุด (Longest Matching) วิธีการนี้ถือเป็นวิธีทางฮิวริสติก (Heuristic) อันหนึ่งซึ่งวิธีนี้จะสแกนประโยคจากซ้ายไปขวา เทียบกับพจนานุกรมเพื่อทำเครื่องหมายกับทุกสายอักขระที่สามารถสร้างเป็นหนึ่งคำให้เป็นจุดย้อนกลับ และเลือกสายอักขระที่ยาวที่สุดเป็นตัวเลือกสำหรับคำแรก ถ้าตัวเลือกนี้สามารถทำให้อัลกอริทึมค้นหาคำที่เหลือได้สมบูรณ์ ตัวเลือกนี้ก็จะเป็น คำแรกจริง ไม่เช่นนั้นอัลกอริทึมก็จะกลับไปยังจุดย้อนกลับที่ทำเครื่องหมายไว้เพื่อใช้เป็นตัวเลือกสำหรับคำแรกใหม่และทำการค้นหาคำที่เหลือต่อไปเป็นเช่นนี้ไปเรื่อย ๆ ตัวอย่างเช่น ประโยค “ รถยนต์มีก๊าซคาร์บอนมอนนอกไซด์ ” ได้ผลของการตัดคำทั้งหมดจะเป็นดังตารางที่ 2.2

ตารางที่ 2.2 ผลการตัดคำโดยวิธีตัดคำให้ยาวสุด (Longest Matching)

ส่วนของคำตัดได้	ส่วนที่เหลือ
รถ	ยนต์มีก๊าซคาร์บอนมอนนอกไซด์
ยนต์	มีก๊าซคาร์บอนมอนนอกไซด์
มี	ก๊าซคาร์บอนมอนนอกไซด์
ก๊าซ	คาร์บอนมอนนอกไซด์
คาร์บอน	มอนนอกไซด์
มอน	นอกไซด์
นอก	ไซด์
ไซด์	

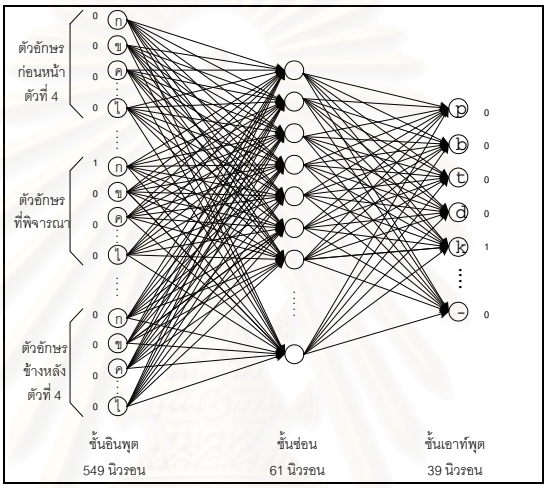
การตัดคำโดยใช้พจนานุกรมนั้นจะหาขอบเขตของหน่วยคำในข้อความที่ต่อเนื่อง ดังนั้น ถ้าหากเก็บทุกคำที่มีอยู่ในภาษาลงในพจนานุกรมทั้งหมด จากนั้นก็ค้นหาและเปรียบเทียบหา คำศัพท์นั้นๆ ว่ามีอยู่ในพจนานุกรมหรือไม่ เพียงเท่านั้นก็จะสามารถหาขอบเขตของคำแต่ละคำได้ แต่ในความเป็นจริงแล้วทำได้ยากมาก หรืออาจจะเป็นไปไม่ได้ที่จะบรรจุคำทุกคำลงใน พจนานุกรมได้ทั้งหมดโดยเฉพาะส่วนที่เป็นวิสามานยนาม (คำนามที่เป็นชื่อเฉพาะ) หรือคำที่เกิด ขึ้นจากการบัญญัติขึ้นมาใหม่ซึ่งไม่สามารถจะคาดการณ์ล่วงหน้าได้ ดังนั้นการตัดคำถึงแม้ว่าจะ อาศัยการเปรียบเทียบคำจากพจนานุกรมก็ตาม ก็จำเป็นที่จะต้องยอมให้มีคำที่ไม่ได้บรรจุไว้ใน พจนานุกรมเกิดขึ้นได้เช่นกัน คำที่บรรจุในพจนานุกรมไม่จำเป็นที่จะต้องเป็นหน่วยคำที่ย่อยที่สุด ที่ความหมายไว้เสมอไป อาจเป็น คำประสม เช่น แม่น้ำ ดูแล ช่างทอง เป็นต้น หรือ วลี เช่น แสงอาทิตย์ หนีเสือปะจระเข้ เป็นต้น

สำหรับอัลกอริธึมการตัดคำด้วยวิธีตัดคำให้ยาวที่สุดที่จะใช้ในวิทยานิพนธ์นี้ จะใช้แนวความคิดและต้นแบบโปรแกรมการตัดคำมาจากศูนย์อิเลคทรอนิคส์และคอมพิวเตอร์แห่งชาติ (Nectec)

2.4 การเข้ารหัสคำทับศัพท์

จากงานวิจัยของ ทัศวรรณ ศูนย์กลาง [10] ซึ่งทำการเข้ารหัสคำเพื่อแปลงคำทับศัพท์ ทั้งที่อยู่ในรูปคำไทยและคำอังกฤษให้อยู่ในรูปรหัสคำอ่านรูปแบบเดียวกัน เพราะหากเป็นคำ ทับศัพท์ที่ตรงกันของทั้งสองภาษาจะอ่านออกเสียงได้เหมือนหรือคล้ายคลึงกัน รหัสคำของคำ ทับศัพท์ต่างๆ ในเอกสารจะถูกสร้างขึ้นในขั้นตอนการสร้างดัชนีในระบบการจัดเก็บสารสนเทศ เมื่อผู้ใช้ป้อนข้อ

คำถามที่เป็นคำทับศัพท์ ระบบค้นคืนก็จะสร้างรหัสคำของคำต่าง ๆ ใน ข้อคำถามเพื่อใช้ค้นหากับรหัสคำที่จัดเก็บไว้ในดัชนี ขั้นตอนวิธีการเข้ารหัสคำจะใช้ แบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก (Backpropagation Neural Networks) ซึ่งเป็นวิธีการเรียนรู้ ของเครื่องที่เหมาะสม สำหรับการนำไปใช้ในการแก้ปัญหาที่เกี่ยวข้องกับการจำแนกหรือแบ่ง ประเภทมาช่วยในการเรียนรู้ รหัสคำอ่านของคำทับศัพท์โดยจะใช้นิวรอลเน็ตเวิร์กให้เรียนรู้ การสร้างรหัสคำอ่านทั้งหมด 4 ชุด สำหรับ (1) คำไทย (2) คำอังกฤษทับศัพท์ คำไทย และ (3) คำอังกฤษ (4) คำไทยทับศัพท์คำอังกฤษ โครงสร้างของนิวรอลเน็ตเวิร์กแสดงในรูปที่ 2.7



รูปที่ 2.7 ตัวอย่างโครงสร้างแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กที่ใช้ในการเรียนรู้คำไทย (ในกรณีคำอังกฤษทับศัพท์คำไทย) ซึ่งมีอินพุตเป็น (_ , _ , _ , _ , ก , น ,ก , พ , ี) และมีเอาท์พุตเป็น 'k'

จากรูปที่ 2.7 โครงสร้างเน็ตเวิร์กประกอบด้วย 3 ชั้นดังนี้

<u>ชั้นอินพุต</u> (Input layer) ประกอบด้วยจำนวนนิวรอนเท่ากับจำนวนอักขระทั้งหมด ของภาษาที่พิจารณาคูณด้วยจำนวนตัวอักษรทั้งหมดที่ใช้พิจารณา (กำหนดให้เป็นค่าคงตัว m) ดังนั้นกรณีคำอังกฤษ ข้อมูลเข้าจะมี 26 x m นิวรอน กรณีคำไทย ข้อมูลเข้าจะมี 61 x m นิวรอน โดยค่า m ซึ่งคือจำนวนตัวอักษร ทั้งหมดที่ใช้พิจารณานั้นมีค่าเป็น 9 เนื่องจากคำไทยมีการใช้ สระที่เกิดจากการใช้ตัวอักษรตั้งแต่ 2 ตัว ขึ้นไป เช่น สระเอีย ดังนั้นจึงควรพิจารณาตัวอักษร ข้างเคียงด้วย เช่น คำว่า "เสถียร" เมื่อตัวอักษร ที่กำลังพิจารณา คือ "เ" ซึ่งจะรู้ว่าเป็นส่วนประกอบ ของสระเอียก็โดยพิจารณาจากตัวอักษรข้างหลัง 4 ตัว หรือเมื่อตัวอักษรที่กำลังพิจารณา คือ "ย" จะรู้ได้ก็โดยพิจารณาจากตัวอักษรข้างหน้า 4 ตัว จากงานวิจัยที่ผ่านมาได้ทำการทดสอบเพื่อหา จำนวนตัวอักษรข้างเคียงที่ใช้พิจารณาร่วมแล้วได้ว่า จำนวนที่ให้ผลในการเรียนรู้ที่ดี คือพิจารณา ตัวอักษรข้างหน้า 4 ตัว ข้างหลังอีก 4 ตัวรวมกับตัวที่ กำลังพิจารณาอีก 1 ตัว

ชั้นซ่อน (Hidden layer) ได้ทำการทดลองเพื่อหาจำนวนนิวรอนที่เหมาะสม (โดยได้จากค่าที่ใช้ในการฝึกแล้วให้ผลการเรียนรู้ดีที่สุด) สำหรับแต่ละเน็ตเวิร์ก ซึ่งสำหรับคำไทยจะมี 61 นิวรอน ส่วนคำอังกฤษจะมี 234 นิวรอน

ตารางที่ 2.3 รหัสเสียงสำหรับคำไทยทับศัพท์คำอังกฤษ

เสียงพยัญชนะ		รหัสเสียง	เสียงสระ		รหัสเสียง
ไทย	อังกฤษ		ไทย	อังกฤษ	
พ	p	p	◌ี	–ee, –ei, –ea,ey	E
บ	b	b	◌ิ	i	I
ท,ต	t, th	t	เ	e,–ay	e
ด	d, th	d	แ	a, –air, –are	w
ก,ค	c, k, g	k	–อ	a,o	$
ช	ch, sh	c	ออ	a,–aw,au	@
จ	j, ch, g	j	◌ุ	u	u
ฟ	f, ph	f	◌ู	–oo	U
ว	w, v	v	◌ั	u	V
ส,ซ	s, z	s	เ–อ	–ur,er,–ir	W
ฮ	h	h	◌ะ	a	a
ม	m	m	โ	–ome,o	o
น	n	n	ไ, ใ, ◌ัย,–าย	ie,ai	!
ง	ng	g	–าว	–ow,ou, our	R
ล	l	l	–วย	oi	O
ร	r	r	เ◌ีย	–ear,ia	I
ย	y	y	◌ัว	–our,ua	Y
ตัวอักษรที่ไม่ออกเสียง		_	เ–า	ou,au	x
			เ◌ิ	–or	q
			◌ิว	–ew,eua	X
			เ◌ิล	–le	Q

<u>ชั้นเอาท์พุต</u>(Output layer) จะมีจำนวนนิวรอนเท่ากับรหัสเสียงพยัญชนะและเสียงสระ ที่เป็นไปได้ทั้งหมด ซึ่งในกรณีคำไทยและคำอังกฤษทับศัพท์คำไทยจะมี 35 นิวรอน และในกรณี คำอังกฤษและคำไทยทับศัพท์คำอังกฤษจะมี 39 นิวรอน (ดูตารางที่ 2.3 , ตารางที่ 2.4 และตารางที่ 2.5 ประกอบ)

ตารางที่ 2.4 รหัสเสียงพยัญชนะสำหรับคำอังกฤษทับศัพท์คำไทย

เสียงพยัญชนะ		รหัสเสียง	
ไทย	อังกฤษ	ตัวต้น	ตัวสะกด
ก ข ค ฆ	ck, g, k, x, c, kh, q	k	k
ง	ng	g	g
จ ฉ ช ฌ	j, ch, x	c	t
ซ ส ศ ษ สร ทร	s, z	s	t
ญ ย หย หญ	y	y	n
ด ฎ ฑ	d	d	t
ต ฏ ถ ฐ ท ฑ ธ ฒ	t, th, dh	t	t
ณ น หน	n	n	n
บ	b	b	p
ป ผ พ ภ	p, ph, bh	p	p
ฝ ฟ	f	f	p
ม	m	m	m
ร ฤ	r	r	n
ล ฬ ฦ	l	l	n
ว	w, v	v	–
ห ฮ	h	h	–
ตัวอักษรที่ไม่ออกเสียง		_	_

ในการฝึก (Train) นิวรอลเน็ตเวิร์กให้เรียนรู้การสร้างรหัสคำอ่านนั้น ข้อมูลที่ใช้ฝึกได้มาจากการนำความรู้ทางภาษาศาสตร์ ได้แก่ หลักเกณฑ์ในการถอดอักษร หลักการถ่ายเสียง หลักการอ่านออกเสียง ทั้งภาษาไทยและภาษาอังกฤษมาสร้างเป็นตารางรหัสเสียง

ตารางที่ 2.5 รหัสเสียงสระสำหรับคำอังกฤษทับศัพท์คำไทย

เสียงสระ		รหัสเสียง
ไทย	อังกฤษ	
◌ิ ◌ี	i, ee	i
ะ า ◌ั	a, u, ar	a
เ–ะ เ–	e	e
แ–ะ แ–	ae	w
◌ึ ◌ื ◌ือ◌ุ ◌ู	u,eu,ue,eo,oo	u
โ–ะ โ– เ–าะ –อ	o	o
เ–อะ เ–อ เ◌ิ	er, oe	W
เ◌ียะ เ◌ีย	ia, ie, aiu	I
เ◌ือะ เ◌ือ ◌ัวะ ◌ัว –ว–	ua, ue, ea, ui	Y
ไ– ใ– ไ–ย ◌ัย –าย	ai, ie, uy	!
เ–า –าว	ao, ou, ow	R
โ–ย –อย	oi, oy	x
◌ิว	iu	X
เ–ว	eo	q
เ–ย	oei	Q
เ◌ือย –วย	uai, uay, ou	O
แ–ว	aeo, eo, aew	$
เ◌ียว	ieo, eaw, eo, ew, iow, iau, iew, iaw	@

สำหรับคำไทยทับศัพท์คำอังกฤษ (ดังในตารางที่ 2.3) และตารางรหัสเสียงสำหรับคำอังกฤษ ทับศัพท์คำไทย (ดังในตารางที่ 2.4 และ 2.5) ซึ่งรหัสคำอ่านจะประกอบด้วยสองส่วน คือ ส่วนที่เป็นเสียงพยัญชนะและส่วนที่เป็นเสียงสระและสำหรับการฝึกคำไทยจะต้องนำคำเหล่านี้มาประมวลผลตัวอักษรเบื้องต้นก่อน ได้แก่ การตัดไม้ไต่คู้ วรรณยุกต์ การันต์และอักษรควบที่มี ตัวการันต์ออกเพราะตามหลักการถอดอักษรไทยเป็นอังกฤษจะไม่พิจารณาตัวอักษรเหล่านี้ [4] จากนั้นจึงนำคำไทยที่ได้หลังผ่านกระบวนการนี้ส่งให้นิวรอลเน็ตเวิร์กเรียนรู้ ส่วนคำอังกฤษ สามารถส่งไปเรียนรู้ได้เลย หลังจากทำการฝึกเสร็จก็จะนำน้ำหนัก (weight) ของแต่ละเน็ตเวิร์ก ที่ให้ผลการเรียนรู้ที่ดีที่สุดมาใช้ในการเข้ารหัสคำ เมื่อได้รหัสคำอ่านจากนิวรอลเน็ตเวิร์กแล้ว จะทำการตัดรหัสที่ไม่ออกเสียง (_) ออกและทำการย้ายรหัสเสียงสระไปต่อท้ายรหัสเสียง พยัญชนะ ในการฝึก เช่นคำว่า “กนกพีระวุฒิ” เป็นรหัสคำ “kanokpiravut_” จะฝึกโดยทำการเลื่อนคำ ไปทีละ หนึ่งตัวอักษร โดยในครั้งแรกจะฝึกด้วย (_ , _ , _ , _ , **ก** , น ,ก , พ , ี) → k ซึ่งในชั้นอินพุตที่ “ก” ของอักษรที่พิจารณา จะถูกกำหนดให้มีค่าเป็น 1 ในชั้นเอาท์พุตที่ “k” ของรหัสเสียงจะถูกกำหนดให้ มีค่าเป็น 1 ส่วนที่เหลือจะมีค่าเป็น 0 ในการทดสอบเอาท์พุตที่มีค่ามากที่สุดจะถูกเลือก ส่วนในกรณีของคำไทยที่มีการให้เสียงสระ ลดรูป ทำให้บางครั้งเอาท์พุตอาจมี 2 เราจะเลือกจำนวนเอาท์พุตว่าเป็น 1 หรือ 2 โดยนำเอาท์พุต 2 ตัวที่มีค่ามากที่สุดมาหาผลต่างและเมื่อผลต่างมีค่าไม่เกินค่า threshold จะได้ว่ามีจำนวน เอาท์พุตเป็น 2 ตัว จากการทดลองได้ทำการคำนวณหาค่า threshold จนได้ค่าที่เหมาะสม คือ 0.3

<u>ตัวอย่าง</u> ต้องการเข้ารหัสคำว่า “กนกพีระวุฒิ” ซึ่งเป็นคำไทย มีขั้นตอนการทำงานดังนี้

1. ประมวลผลตัวอักษรเบื้องต้น จะได้ กนกพีระวุฒิ → กนกพีระวุฒิ เช่นเดิมสร้างเป็นอินพุตเพื่อส่งให้นิวรอลเน็ตเวิร์ก โดยพิจารณาตัวอักษรทั้งหมดครั้งละ 9 ตัว โดยตัวที่สนใจคือตัวที่ 5 และพิจารณาตัวอักษรข้างเคียงข้างหน้า 4 ตัวและข้างหลังอีก 4 ตัว

   (_ , _ , _ , _ , **ก** , น , ก , พ , ี) → k
   (_ , _ , _ , ก , **น** , ก , พ , ี , ร) → a,n
   (_ , _ , ก , น , **ก** , พ , ี , ร , ะ) → o,k
   (_ , ก , น , ก , **พ** , ี , ร , ะ , ว) → p
   (ก , น , ก , พ , **ี** , ร , ะ , ว , ุ) → i
   (น , ก , พ , ี , **ร** , ะ , ว , ุ , ฒ) → r
   (ก , พ , ี , ร , **ะ** , ว , ุ , ฒ , ิ) → a
   (พ , ี , ร , ะ , **ว** , ุ , ฒ , ิ , _) → v

(◌ิ , ร , ะ , ว , ◌ุ , ฒ , ◌ิ , _ , _) → u

(ร , ะ , ว , ◌ุ , ฒ , ◌ิ , _ , _ , _) → t

(ะ , ว , ◌ุ , ฒ , ◌ิ , _ , _ , _ , _) → _

ตัดรหัสที่ไม่ออกเสียง (_) ออกและทำการย้ายรหัสเสียงสระไปต่อท้ายรหัสเสียงพยัญชนะ

kanokpiravut_ → kanokpiravut → knkprvtaoiau

ขั้นตอนการค้นคืน

รหัสคำที่ได้ของคำคู่ที่ตรงกันทั้งสองภาษาอาจจะไม่ตรงกันทุกตัวอักษรแต่จะมีลักษณะคล้ายกัน เนื่องจากหลักเกณฑ์การทับศัพท์ที่ใช้ในปัจจุบันมีหลายรูปแบบเพื่อให้ได้ค่าความเที่ยง (Precision) และค่าเรียกคืน (Recall) ที่ดี จะใช้การเปรียบเทียบรหัสคำแบบประมาณ (Approximate matching) ซึ่งอาศัยการคำนวณความแตกต่างของรหัสคำด้วยเทคนิคระยะ แก้ไข สั้นที่สุด (Minimal Edit distance) โดยคำนวณหาจากจำนวนครั้งที่น้อยที่สุดที่ใช้ในการเพิ่ม การลบ และการแทนที่ แต่ละตัวอักขระ เพื่อทำให้รหัสคำทั้งสองเหมือนกัน

การคำนวณความแตกต่างนี้อาศัยเทคนิคกำหนดการพลวัต (Dynamic programming) ซึ่งวิธี การคำนวณสามารถเขียนให้อยู่ในรูปการคำนวณด้วยความสัมพันธ์เวียนเกิด Edit (P_j, W_k) ดังนี้ [4]

$$\text{Edit}\ (P_0, W_0) = 0$$

$$\text{Edit}\ (P_j, W_0) = j$$

$$\text{Edit}\ (P_0, W_k) = k$$

$$\text{Edit}\ (P_j, W_k) = \min\{\ \text{Edit}\ (P_{j-1}, W_k) + 1,\ \text{Edit}\ (P_j, W_{k-1}) + 1,\ \text{Edit}\ (P_{j-1}, W_{k-1}) + r(p_j, w_k)\ \}$$

โดยที่ $P_j = p_1\ p_2\ p_{3...}\ p_j$ เป็นสายอักขระต้นแบบ มีความยาว j ตัวอักษร

$W_k = w_1\ w_2\ w_{3...}\ w_k$ เป็นสายอักขระเป้าหมาย มีความยาว k ตัวอักษร

$r(p_j, w_k) = 0$ ถ้า p_j เท่ากับ w_k

$= 1$ ถ้า p_j ไม่เท่ากับ w_k

ถ้าความแตกต่างที่ได้มีค่าไม่เกินกว่าค่าที่ยอมรับได้ (ค่าคงที่ d) จะสรุปได้ว่ารหัสคำทั้งสอง เป็นรหัสที่มาจากคำหลักที่ตรงกันในอีกภาษา

<u>ตัวอย่าง</u> ต้องการทดสอบคำว่า “kanokpeerawut” และ “กนกพีระวุฒิ” เป็นคำทับศัพท์ที่ตรงกันในภาษาไทย–อังกฤษหรือไม่ โดยทำการเข้ารหัสคำแล้วคำนวณหาค่า ความแตกต่างได้ดังต่อไปนี้

<u>การเข้ารหัสคำ</u>

kanokpeerawut	กนกพีระวุฒิ
⬇	⬇
kanokpi_ravut	konkopiravut_
⬇	⬇
knkprvtaoiau	knkprvtooiau

<u>การค้นคืน</u>

Edit (knkprvtaoiau, knkprvtooiau) = 1

จากตัวอย่าง รหัสคำทั้งสองมีค่าความแตกต่างเป็น 1 ถ้าเรากำหนดให้ d=1 ก็จะสามารถค้นคืน “kanokpeerawut” จาก “กนกพีระวุฒิ” ได้

2.5 อัลกอริทึมการจัดกลุ่มข้อมูล

การจัดกลุ่มข้อมูลมีหลายแบบ แต่ละแบบมีคุณลักษณะแตกต่างกันไปเช่น Hierarchical agglomerative Clustering (HAC) , K-Means , Buckshot , Fractionation ,Single-Pass และ Suffix Tree Clustering (STC) เป็นต้น

2.5.1 Hierarchical Agglomerative Clustering (HAC) [5]

HAC เป็นอัลกอริทึมที่ได้รับความนิยมใช้มากในด้าน ระบบ GIS,Molecular biology, อิมเมจโปรเซสซิ่ง และดาต้ามายนิ่ง ลักษณะการทำงานของอัลกอริทึมแสดงได้ดังนี้

```
//AGGLOMERATIVE (BOTTOM-UP) VERSION
Place each object in its own cluster;
While (halting criterion OR one cluster)
 {
        Merge the most similar clusters basing on some measure;
}
//DIVISIVE (TOP-DOWN) VERSION
Place all objects in a single cluster;
```

```
While (halting criterion OR each object in its own cluster)
{
        Split the most dissimilar cluster basing on some measure;
}
```

รูปที่ 2.8 การทำงานของ Hierarchical Agglomerative Clustering

หลักการของวิธีนี้คือ ใช้กฎในการการวัดความคล้ายกันของเอกสาระหว่างกลุ่มข้อมูล (Cluster) และกฎในการรวมหรือแยกเอกสารเข้าในกลุ่มข้อมูล โดยมีการกำหนดเงื่อนไขการหยุด (Halting criterion) ซึ่งเป็นสาเหตุที่ให้อัลกอริทึมหยุด ทำงาน

ข้อดี

วิธีนี้สร้างโครงสร้างข้อมูลแบบลำดับขั้น ณ ขั้นตอนการประมวลผลเรียกว่า dendrogram ซึ่งแสดงให้เห็นอย่างชัดเจนถึงการสร้างกลุ่มข้อมูล และยอมให้ค้นหาผ่านโครงสร้างผลลัพท์แบบลำดับขั้น กลุ่มข้อมูลที่สร้างมีประสิทธิภาพที่ดี

ข้อเสีย

- ใช้เวลานาน ซึ่งเท่ากับ $O(n^2)$
- เอกสารที่ถูกจัดรวมอยู่ในกลุ่มเดียวกันไม่สามารถแยกออกไปอยู่กลุ่มข้อมูลอื่นได้อีก
- ไม่เหมาะกับการจัดกลุ่มข้อมูลขนาดใหญ่หรือปริมาณมาก

2.5.2 K-means [5]

ไม่มีการสร้างลำดับชั้นของผลลัพท์เหมือน HAC แต่จะแสดงผลลัพท์ทั้งหมดในระดับเดียวกัน ลักษณะการทำงานของอัลกอริทึมแสดงได้ดังนี้

```
Partition objrcts into k non-empty subsets;
Do {
    Compute centroids of cluster of the current partition;
    Assign each object to the cluster with the nearest centroid;
} while (no reassignments have been made);
```

รูปที่ 2.9 การทำงานของ K-Means

K-means เริ่มต้นด้วยการคำนวนค่าจุดกึ่งกลาง (centroid) ของแต่ละกลุ่มข้อมูล แล้วสุ่มกลุ่มข้อมูล เพื่อทำการเปรียบเทียบค่าของเอกสารทีละตัวภายในกลุ่มข้อมูลนั้นกับค่า จุดกึ่งกลาง (centroid) ที่คำนวณไว้แล้วว่าเอกสารนั้นมีค่าใกล้เคียงกับจุดกึ่งกลาง (centroid) ใด แล้วจึงจะทำการย้ายเอกสารนั้นไปอยู่ภายใต้กลุ่มข้อมูลนั้น วิธีนี้ใช้เวลา O(tkn)

เมื่อ n คือ จำนวนของเอกสาร

k คือ จำนวนกลุ่มข้อมูล

t คือ จำนวนรอบที่มากที่สุดเพื่อทำให้เอกสารทุกตัวถูกจัดกลุ่มให้อยู่ภายใต้กลุ่มข้อมูลที่เหมาะสม

<u>ข้อดี</u>

การแบ่งส่วนของข้อมูลมาก่อนช่วยให้การจัดกลุ่มข้อมูลเร็วยิ่งขึ้น

<u>ข้อเสีย</u>

- มักจะหยุดการทำงานที่ local optimum สำหรับการหา global optimum ต้องอาศัยเทคนิคเช่น deterministic annealing หรือ อัลกอริทึมเชิงพันธุกรรม (genetic algorithm)
- ต้องทราบจำนวนกลุ่มข้อมูลก่อน
- ไม่สามารถจัดการข้อมูล noisy data และ outliers
- ไม่เหมาะสมกับการจัดกลุ่มข้อมูล non-convex shapes

2.5.3 **ตัวแยกแยะแบบเบย์** (Bayesian classifiers) [5]

วิธีนี้ซับซ้อนกว่าอัลกอริทึมแบบ k-means และให้ผลลัพท์ที่ดีกว่าเพราะสามารถสร้างกลุ่มข้อมูลที่มีขนาดต่างกันได้ นิยมใช้ในปัญหาการจำแนกประเภท (Classification Problem) แต่ไม่นิยมใช้ในการจัดกลุ่มข้อมูล

2.5.4 **เวคเตอร์สเปซ** (Vector Space) [5]

คือการเลือกคำสำคัญ k ตัวแรกจากกลุ่มของเอกสารที่เป็นฐานสร้างเป็นเวคเตอร์ขนาด k ลำดับและสร้างเวคเตอร์ขนาด k ของแต่ละเอกสารให้มีลำดับที่ตรงกับคำสำคัญ k ตัว โดยการให้ค่าลำดับที่ i เป็น 1 ถ้าคำสำคัญลำดับที่ i ปรากฏในเอกสารนั้นและมีค่าเป็น 0 ถ้าไม่ปรากฏ เราสามารถเปรียบเทียบความเหมือนหรือความต่างของ 2 เอกสารได้โดยใช้ หลักการวัดแบบ Jaccard ดังนี้

$$d(s1,s2) = \frac{\sum_i \min(s1i,s2i)}{\sum_i \max(s1i,s2i)} \quad \ldots\ldots\ldots (1)$$

สังเกตุได้ว่าคำ ๆ หนึ่งในภาษาส่วนใหญ่สามารถเขียนได้หลายรูปแบบเช่น “fence” ซึ่งเหมือน “fencing” ในประโยคต่าง ๆ ณ ที่นี้เราต้องถือว่าเป็นคำ ๆ เดียวกัน ดังนั้นหลังจากที่กำจัดคำทั่วไปออกไปแล้ว ต้องมีการทำอัลกอริทึมสเต็มมิ่ง (stemming algorithm) เพื่อแปลงคำที่เขียน ในรูปแบบ แตกต่างกันแต่มี รากศัพท์เดียวกันให้อยู่ในรูปแบบเดียวกันก่อน

2.5.5 **เอนแกรม** (N-Gram) [5]

คือการใช้หลักพิจารณาความยาวขนาด n ตัวอักษรจากเอกสาร ยกตัวอย่างเช่น ให้ n=5 และมีเอกสารที่มีข้อความ “…relational fuzzy clustering…” ค่า N-gram ที่ได้คือ “relat” “elati” “latio” และ “ation” วิธี N-Gram นี้ไม่ต้องใช้วิธีการของสเต็มมิ่ง (stemming) เข้ามาช่วย โดยสามารถเปรียบเทียบความเหมือนหรือความต่างของ 2 เอกสารได้ โดยใช้หลักการวัดสัดส่วนของ N-gram ที่ปรากฏในแต่ละเอกสารด้วยการคำนวณแบบ Dice coefficient

$$\frac{2*C}{(A+B)} \quad \ldots\ldots\ldots\ldots (2)$$

โดยที่ AและB หมายถึง จำนวน N-gram จากคำอธิบายย่อจำนวนหนึ่งที่ได้รับกลับมาจากระบบค้นหาข้อมูล(snippet) และ

C หมายถึง จำนวนของเอกสาร

2.5.6 **การจัดกลุ่มข้อมูลแบบซัฟฟิกทรี** (Suffix Tree Clustering -STC) [5]

ในงานวิจัยชิ้นนี้เลือกใช้การจัดกลุ่มข้อมูลแบบ STC ซึ่งเป็นลิเนียร์อัลกอริทึม (linear) ที่ใช้วลี (Phrase) ซึ่งอาจเป็นคำคำเดียวหรือเป็นกลุ่มคำภายในเอกสารมาเป็นหัวข้อในการจัดกลุ่ม ข้อมูลและคำนึงถึงคำข้างเคียงและลำดับของคำ ซึ่งช่วยให้การค้นหาข้อมูลมีประสิทธิภาพที่ดีกว่าการพิจารณาเปรียบเทียบหาคำสำคัญเพียงอย่างเดียว

การจัดกลุ่มข้อมูลลักษณะนี้อาศัยทฤษฎีโครงสร้างข้อมูลต้นไม้ (Tree) ซึ่งมีคำอธิบายเรื่องคำศัพท์ นิยามของต้นไม้แบบต่าง ๆ และอัลกอริทึมที่เกี่ยวข้องดังแสดงในภาคผนวก ง

```
Split text into sentences consisting of words;
/* Phrase 1. Creation of a Generalized Suffix Tree of all sentences */
for each document {
        for each sentence {
                apply stemming to sentence words;
         /* A word is a stop-term if it exists in predefined list,exists in more than a
         *  certain percent of input documents or in less than a certain number of
         *  input documents. Stop terms can be ignored during clustering */
                if (sentence length > 0 ) {
                        insert sentence and all its substring into generalized
                        suffix tree update internal nodes with the index to current
                        document while rearranging the tree;
                } /*if*/
        } /* for each sentence */
} /* for each document */
/* Phrase 2. Build a list of base clusters */
for each node in the tree {
        if (number of documents in node's subtree > 2) {
                if (candidate_base_cluster_score > minimal_base_cluster_score) {
                        add a base cluster to the list of base clusters;
                }
        }
} /* for each node in the tree */
/* Phrase 3. Merge base clusters */
Build a graph where nodes are base cluster ans there is a link between node A and B
If and only if the number of common documents indexed by A and B is greater than
The MERGE_THRESHOLD;
Clusters are coherent subgraphs of that graph;
```

รูปที่ 2.10 อัลกอริทึมการจัดกลุ่มข้อมูลแบบ Suffix Tree Clustering

สามารถอธิบายขั้นตอนการจัดกลุ่มข้อมูลแบบ STC (Suffix Tree Clustering) ดังแสดงในรูปที่ 2.10 ซึ่งมี 3 ขั้นตอนหลักคือ

1. Document parsing

   1.1 เอกสารต่าง ๆ จะถูกแปลงรูปออกเป็นลำดับชุดของคำและมีการกำหนดขอบเขต ของแต่ละคำนั้นโดยใส่เครื่องหมายพิเศษคั่นระหว่างคำเช่น จุด (.) หรือ คอมมา (,) หรือ เซมิโคลอน (;) หรือ เครื่องหมายคำถาม (?) หรือ HTML tags เช่น <p> ,
 , <li> หรือ <td> เป็นต้น คำที่อยู่ภายในเครื่องหมายคำพูดหรือวงเล็บจะถือว่าเป็นประโยคที่ไม่ ต้องการ

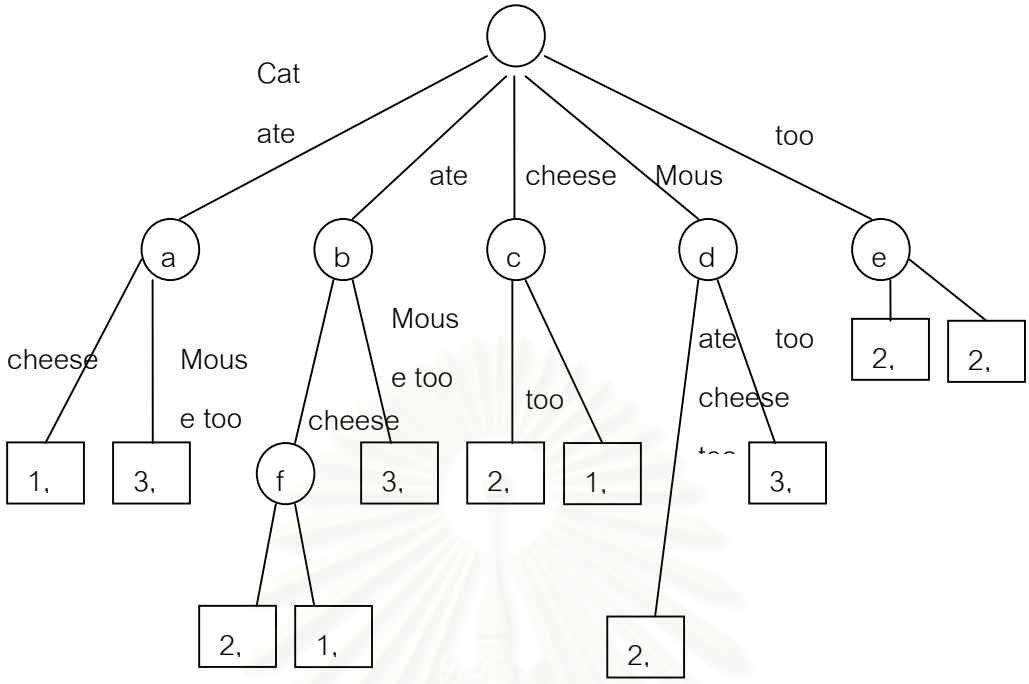
   1.2 ตัดคำจำพวกคำทั่วไป (Stop word) คือคำที่ไม่เกี่ยวข้องกับข้อมูลที่ต้องการ เครื่องหมายวรรคตอนหรือคำทั่วไปออกเช่น ตัวเลขหรือ HTML tags เป็นต้น

   1.3 กลุ่มคำในแต่ละเอกสารจะถูกแปลงโดยการทำสเต็มมิ่งอัลกอริทึม(Stemming Algorithm) คือ การลบคำนำหน้า (prefix) ,ลบคำต่อท้าย (suffix) และแปลงคำพหูพจน์เป็นคำ เอกพจน์ (ในกรณีที่เป็นคำภาษาอังกฤษ)

2. สร้างกลุ่มข้อมูลฐานหรือเบสคลัสเตอร์ (Base Clustering) โดยการใช้ซัฟฟิกทรี (Suffix Tree) คือการทำดัชนีย้อนกลับ (inverted index) ของแต่ละวลีภายในเอกสารทั้งหมดที่ได้จากการค้นหาจากระบบค้นหาข้อมูล ซึ่งจะสร้างขึ้นมาทันทีที่เริ่มต้นอ่านเอกสารแต่ละฉบับโดยใช้โครงสร้างข้อมูลที่เรียกว่าซัฟฟิกทรี (Suffix Tree) เวลาที่ใช้ในการสร้างเบสคลัสเตอร์แปรผันโดยตรงกับจำนวนของเอกสารทั้งหมด กล่าวคือถ้าจำนวนของเอกสารมากก็จะใช้ เวลาในการสร้างคลัสเตอร์มากนั่นเอง ยกตัวอย่างการสร้างซัฟฟิกทรีของข้อความ S เราจะถือว่าเอกสารคือกลุ่มของคำที่เรียงต่อกันไม่ใช่ตัวอักษรทีละตัว ซัฟฟิกทรีที่สร้างมี คุณสมบัติดังนี้

   - ซัฟฟิกทรีเป็นโครงสร้างต้นไม้ที่มีโหนดหนึ่งเป็นราก(rooted tree) และมีด้านเชื่อมกับ โหนดอื่น ๆ ในต้นไม้โดยที่มีค่ากำกับอยู่บนด้าน (directed tree)
   - แต่ละโหนดภายในต้นไม้มีลูกอย่างน้อย 2 โหนด
   - แต่ละกิ่ง (edge) ถูกระบุด้วยคำที่อยู่ภายในประโยค S และเป็นกลุ่มคำที่ต่อเนื่องมาตามเส้นทางภายใน ต้นไม้ เริ่มต้นจากราก
   - ไม่มีกิ่ง (edge) สองด้านใด ๆ ออกจากโหนดเดียวกันที่จะถูกระบุด้วยคำเริ่มต้นเหมือนกัน
   - แต่ละ Suffix s ของประโยค S จะต้องมี suffix โหนด ซึ่งถูกระบุด้วยค่าของ s

   ยกตัวอย่างการสร้าง suffix tree จากตัวอย่างของชุดเอกสาร “cat ate cheese” “mouse ate cheese too” และ “cat ate mouse too” ได้ดังรูปที่ 2.11



รูปที่ 2.11 ตัวอย่าง suffix tree

จากรูปที่ 2.11 พบว่ามีโหนดภายใน (internal node) ซึ่งจะถูกนำมาพิจารณาเป็นเบสคลัสเตอร์ ดังแสดงในตาราง 2.6

ตาราง 2.6 โหนดภายใน (Internal node) ในต้นไม้จากตัวอย่างของชุดเอกสาร “cat ate cheese” “mouse ate cheese too” และ “cat ate mouse too” ที่เป็นเบสคลัสเตอร์

Node	Phrase	Documents
a	cat ate	1,3
b	ate	1,2,3
c	cheese	1,2
d	mouse	2,3
e	too	2,3
f	ate cheese	1,2

หลังจากสร้างซัฟฟิกทรีแล้ว จะพิจารณากลุ่มข้อมูลฐานที่มีค่าสูงสุด (maximal base cluster) โดยคำนวนจากสูตรโดยประมาณดังนี้

$$s(m) = |m| \cdot f(|m_p|) \cdot \sum tfidf(w_i)$$

เมื่อ $s(m)$ คือ คะแนนของเบสคลัสเตอร์ “m”

$|m|$ คือ จำนวนของเอกสารที่มีเบสคลัสเตอร์ “m” ปรากฏอยู่

w_i คือ คำใน m_p

$tfidf(w_i)$ คือ คะแนนที่คำนวนให้กับคำแต่ละคำใน m_p

$|m_p|$ คือ จำนวนของคำทั้งหมดใน m_p ที่ไม่ใช่คำจำพวกคำทั่วไป (Stop word)

f คือ ฟังก์ชันปรับนำหนักของคำจะแปรผันโดยตรงกับจำนวนคำระหว่าง 1 คำ ถึง 5 คำและมีค่าคงที่เมื่อจำนวนคำตั้งแต่ 6 คำขึ้นไป

คำจำพวกคำทั่วไป (Stop word) จะพิจารณาได้จากรายชื่อคำที่ถูกรวบรวมอยู่ในแฟ้มข้อมูล ทั้งภาษาไทยและภาษาอังกฤษเช่น คำว่า “the” , “there” , “their” , “thus” และ”then” ในภาษาอังกฤษ และ คำว่า “การ” , “หรือ” , “ความ” และ “กัน” สำหรับภาษาไทยเป็นต้น คำที่มีความยาวน้อยกว่า 3 คำ หรือ คำที่มีมากเกินไปในเอกสาร เช่น มากเกิน 40% เมื่อเทียบกับจำนวนคำทั้งหมดในเอกสารนั้น

ฟังก์ชั่น tfidf (Term frequency) ใช้ในการคำนวนนำหนักของคำแต่ละคำ ในเอกสารโดยมีสูตรดังนี้

$$tfidf(w_i, d) = (1+\log(tf(w_i, d))) \cdot \log(1+N/df(w_i))$$

เมื่อ $tf(w_i, d)$ คือ จำนวนครั้งที่คำ w_i ปรากฏในเอกสาร d

N คือ จำนวนของเอกสารทั้งหมด

$df(w_i)$ คือ จำนวนของเอกสารทั้งหมดที่มีคำ w_i ปรากฏอยู่

ตาราง 2.7 ค่าเฉลี่ยของเบสคลัสเตอร์ที่ปรากฏอยู่ในแต่ละเอกสาร

	A	B	C
ate	0.30103	0.30103	0.30103
cat	0.39794	0.39794	0.39794
cheese	0.39794	0.39794	0.39794
mouse	0.39794	0.39794	0.39794
too	0.39794	0.39794	0.39794

เมื่อคำนวนคะแนนของแต่ละเบสคลัสเตอร์แล้ว จะเลือกเอาเบสคลัสเตอร์ที่มีคะแนน มากที่สุดจำนวน N ตัวซึ่งการเลือกค่า N ที่เหมาะสมจะมีผลต่อการจัดกลุ่มของข้อมูล ในขั้นตอน ต่อไป ด้วย

3. การรวมเบสคลัสเตอร์ ที่มีความคล้ายคลึงกันเป็นคลัสเตอร์ (Cluster) เดียวกัน

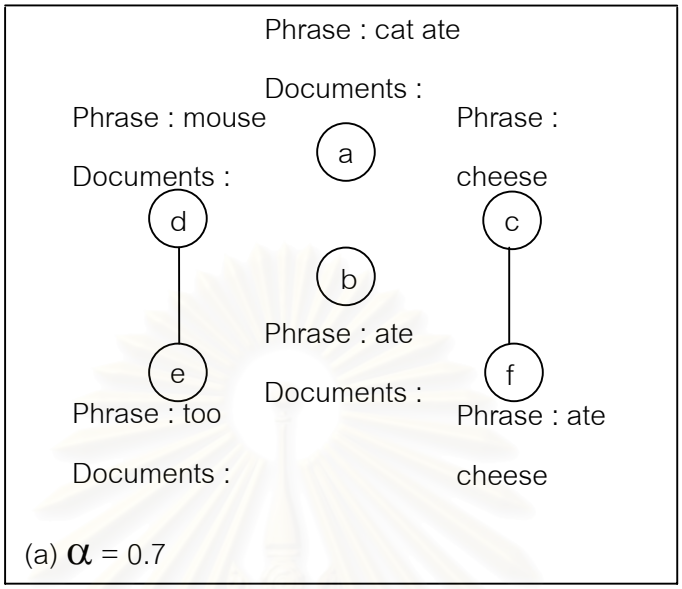
เป็นการรวมเบสคลัสเตอร์ที่ประกอบด้วยเอกสารที่เหมือนหรือคล้ายคลึงกันมากกว่า 50% เข้าเป็นกลุ่มเดียวกัน และแสดงหัวข้อกลุ่มใหม่จากหัวข้อของคลัสเตอร์ที่มารวมกันนั่นเอง การพิจารณาความเหมือนกันของ เบสคลัสเตอร์ต่าง ๆ สามารถคำนวนได้จากฟังก์ชั่นต่อไปนี้

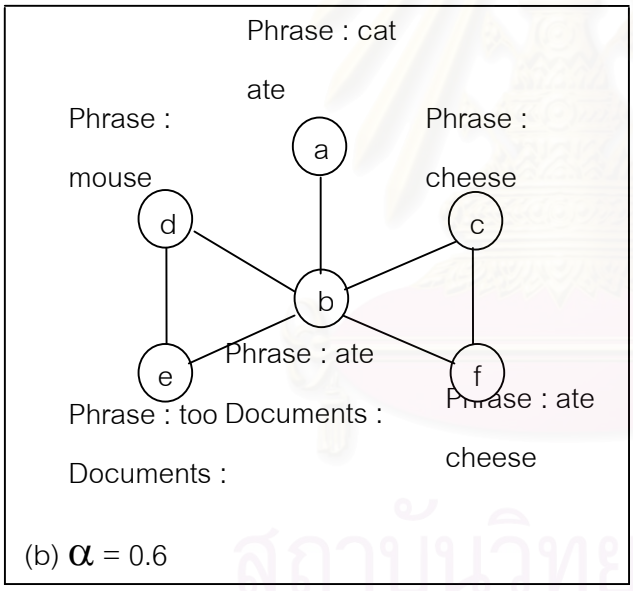
$similarity(m_i , m_j) = 1$ if $| m_i \cap m_j | / | m_i | > \alpha$ and $| m_i \cap m_j | / | mj | > \alpha$

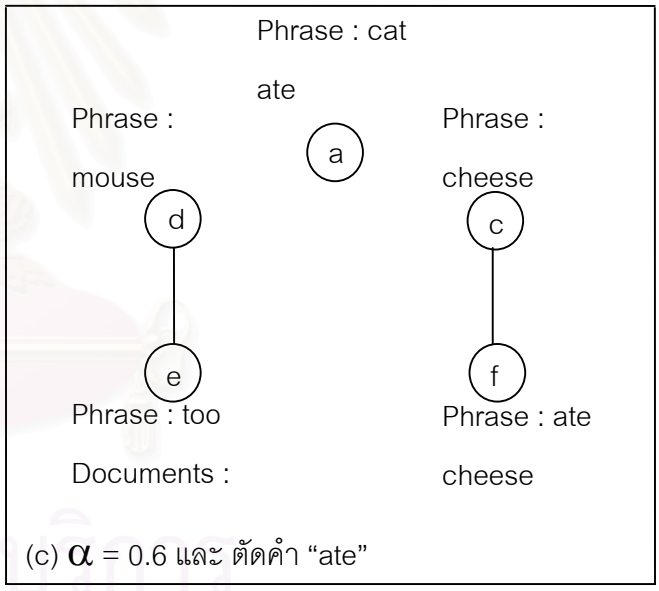
$similarity(m_i , m_j) = 0$ otherwise

เมื่อ α คือ ค่าคงที่ระหว่าง 0 กับ 1

ความสัมพันธ์ระหว่าง เบสคลัสเตอร์จากตาราง 2.6 สามารถแสดงได้ในรูปของกราฟ โดยโหนดต่าง ๆ แสดงถึงเบสคลัสเตอร์ โหนดสองโหนดใด ๆ มีด้านเชื่อมต่อกันก็ต่อเมื่อ เบสคลัสเตอร์ทั้งสองมีค่าความเหมือน (similarity) เท่ากับ 1 และโหนดต่าง ๆ จะถูกรวมกัน เป็นคลัสเตอร์เดียวกันในที่สุด เรียกว่า เมิร์จคลัสเตอร์ (merge cluster)







รูปที่ 2.12 กราฟแสดงความสัมพันธ์ของเบสคลัสเตอร์

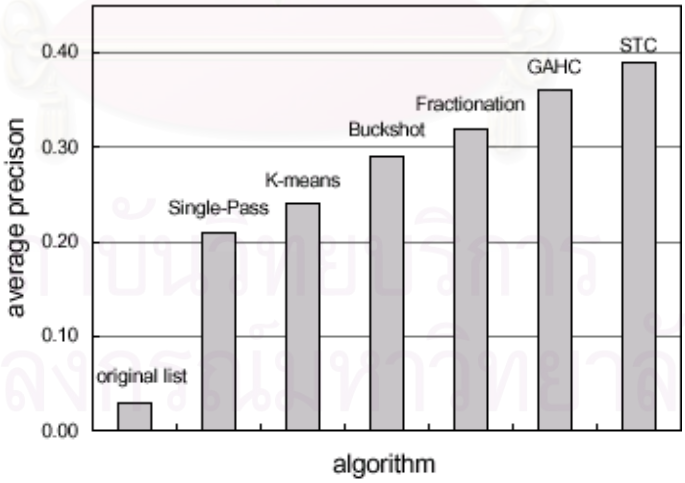
ตาราง 2.8 เมิร์จคลัสเตอร์ที่ได้จากเบสคลัสเตอร์ ของรูปที่ 2.12

รูป	Cluster	Base cluster	**ชุดเอกสาร**
(a)	1	a	1,3
	2	b	1,2,3
	3	d,e	2,3
	4	c,f	1,2
(b)	1	a,b,c,d,e,f,g	1,2,3
(c)	1	A	1,3
	2	d,e	2,3
	3	c,f	1,2

สรุปผลการเปรียบเทียบวิธีการจัดกลุ่มข้อมูลแบบต่าง ๆ

จากผลการวิจัยที่ผ่านมาส่วนใหญ่พบว่า STC ให้ผลลัพธ์ที่มีประสิทธิภาพดีกว่าอัลกอริทึมอื่น ได้มีการทำการทดลองวัดประสิทธิภาพของการจัดกลุ่มข้อมูลแบบต่าง ๆ ดังนี้

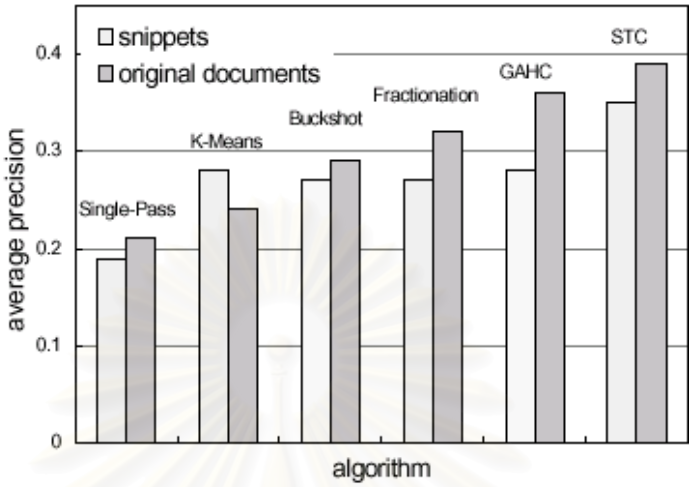
1) ประสิทธิภาพของการค้นคืน



รูปที่ 2.13 ค่าแม่นยำเฉลี่ยที่ได้จากการจัดกลุ่มข้อมูลแบบต่าง ๆ

จากรูปที่ 2.13 พบว่าการจัดกลุ่มข้อมูลแบบ STC ให้ค่าแม่นยำสูงกว่าการจัดกลุ่มแบบอื่น ๆ ผู้ทำการทดลองเห็นว่าเกิดจากการที่ STC พิจารณากลุ่มคำภายในเอกสารและความสัมพันธ์ของคำรวมทั้งยอมให้เอกสารหนึ่ง ๆ สามารถอยู่ภายใต้กลุ่มข้อมูลหลายกลุ่มได้

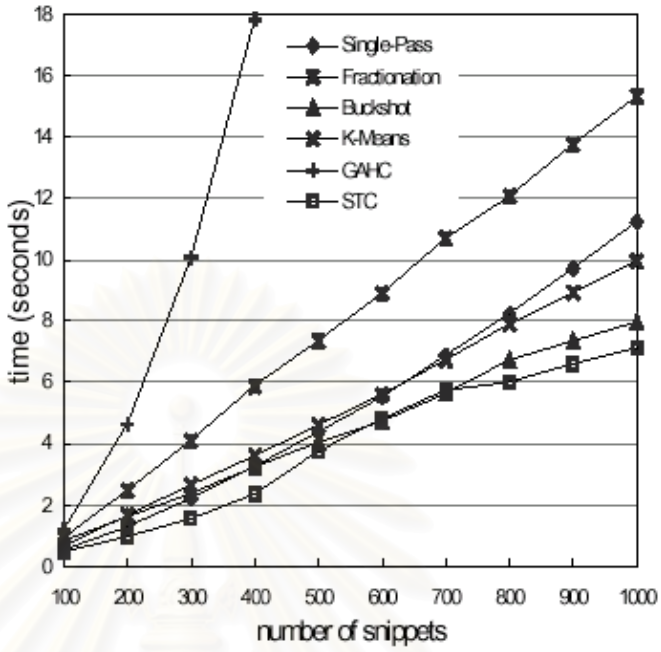
2) ประสิทธิภาพระหว่างการทำงานกับคำอธิบายย่อ (snippet) ของเอกสาร และการทำงานกับเอกสารทั้งหมด



รูปที่ 2.14 ค่าแม่นยำเฉลี่ยที่ได้จากการจัดกลุ่มข้อมูลแบบต่าง ๆ เมื่อทำงานกับคำอธิบายย่อ (snippet) และทำงานกับเอกสารทั้งหมด

จากรูปที่ 2.14 แสดงให้เห็นประสิทธิภาพการทำงานของการจัดกลุ่มข้อมูลแบบต่าง ๆ เปรียบเทียบกับตัวเองเมื่อทำงานกับคำอธิบายย่อ (snippet) ของเอกสารและการทำงาน กับเอกสารทั้งหมด ผลที่ได้แสดงให้เห็นว่าคำอธิบายย่อ (snippet) มีผลให้ค่าแม่นยำลดลง อย่างไรก็ตามค่าที่ลดลงไม่แตกต่างกับการทำงานกับเอกสารทั้งหมดมากนัก

3) เวลาที่ใช้ในการจัดกลุ่มข้อมูล



รูปที่ 2.15 เวลาที่ใช้ในการจัดกลุ่มข้อมูลแบบต่าง ๆ

จากการทดลองที่ผ่านมาพบว่าการจัดกลุ่มข้อมูลสามารถถูกแยกออกจากระบบค้นคืนเป็นอีกส่วนต่างหาก ซึ่งจะรับผลลัพธ์ที่ได้จากการค้นหาของระบบค้นหาและนำมาจัดกลุ่มเพื่อแสดงเป็นผลลัพธ์สุดท้ายแก่ผู้ใช้ต่อไป สำหรับ STC ซึ่งเป็นอัลกอริทึมที่ทำงานแบบ incremental จะรับผลลัพธ์ที่ได้ก่อนหน้านี้มาทำการจัดกลุ่ม ในขณะที่ระบบค้นหาจะค้นหาข้อมูลอื่น ๆ ต่อไป และส่งให้ส่วนการจัดกลุ่ม ทำงานเช่นนี้จนกระทั่งเอกสารสุดท้ายได้รับการจัดกลุ่มข้อมูลเสร็จสิ้น ต่างจากอัลกอริทึมแบบ non-incremental ที่จะรอให้ระบบค้นหาเอกสารจนครบก่อน จึงส่งเอกสารทั้งหมดมาทำการจัดกลุ่มข้อมูลพร้อมกันทีเดียว

2.6 เทคนิคช่วยในการจัดกลุ่มข้อมูล

a) คำทั่วไป (Stop Words)

คำทั่วไป (Stop Word) หรือ คำที่ควรตัดทิ้ง (Stop Term) คือคำที่ใช้ประกอบการสร้างประโยคทั้งในภาษาไทยและภาษาอังกฤษ แต่มิได้มีความหมายสำคัญหรือเป็นใจความสำคัญของประโยค เช่น ประโยค “Amiga was one of the best computer I’ve had” และ “The finest computer I’ve worked on was an Amiga” ซึ่งใช้คำต่างกันแต่ให้ความหมายเดียวกัน คำทั่วไปเหล่านี้ มีผลต่อจำนวนคำในแต่ละเอกสารเพราะเป็นคำทั่วไปที่เป็นส่วนหนึ่งในการสร้าง ประโยค ด้วยเหตุนี้จึงควรกำจัดคำเหล่านี้ออกจากเอกสารก่อนนำไปทำการจัดกลุ่มข้อมูล

ตาราง 2.9 และ 2.10 แสดงตัวอย่าง คำทั่วไปในภาษาไทยและภาษาอังกฤษ ซึ่งคำภาษาอังกฤษทั่วไปไม่มีปัญหาในการตัดทิ้งเนื่องจากการแบ่งคำในภาษาอังกฤษอย่างชัดเจนอยู่แล้ว แต่สำหรับคำภาษาไทยยังมีปัญหาเช่น คำว่า “การ” ซึ่งเป็นคำอาการนาม (คำนาม ที่เกิดจากการ นำ "การ" และ "ความ" นำหน้าคำกริยาหรือคำวิเศษณ์) หรือเป็นคำสามานยนาม (นามทั่วไปไม่ชี้เฉพาะเจาะจง) เช่น การบ้าน การไฟฟ้า ความแพ่ง ความวัวความควาย หรือในบางครั้งคำนี้ก็ถือเป็นส่วนหนึ่งของคำ ๆ หนึ่งเช่น คำว่า “วิชาการ” ดังนั้นในงานวิจัยนี้ การจะถือว่าคำ ๆ นี้เป็นคำทั่วไปหรือไม่ จึงขึ้นอยู่กับประสิทธิภาพการตัดคำของโปรแกรม การตัดคำภาษาไทยที่นำมาใช้ในงานวิจัยด้วย

ตาราง 2.9 ตัวอย่างคำทั่วไป (Stop Word) ภาษาอังกฤษ

about	out	get	is	mine	often	ten	very
by	over	give	it	more	on	than	via
her	own	go	its	moreover	once	that	was
him	same	had	itself	most	one	the	we
his	see	has	keep	mostly	only	their	well
how	seem	hasnt	last	move	or	them	were
however	five	have	my	much	up	themselves	what
hundred	for	he	may	must	us	then	whatever

ตาราง 2.10 ตัวอย่างคำทั่วไป (Stop Word) ภาษาไทย

การ	ความ	และ	หรือ	เป็น	ฉัน	เธอ	เรา
เอง	คุณ	เขา	หลาย	แล้ว	เช่น	แต่	ๆ

b) **สเต็มมิ่งอัลกอริทึม** (Stemming algorithm) [12]

เป็นอัลกอริทึมที่แปลงคำภาษาอังกฤษที่อยู่ในรูปแบบต่าง ๆ ให้กลับเป็นรากคำ เช่นการแปลงคำภาษาอังกฤษที่เขียนในรูปพหูพจน์ให้กลายเป็นคำเอกพจน์ เนื่องจากคำหนึ่งคำ ในภาษาอังกฤษสามารถเขียนได้หลายรูปแบบเช่นเขียนในรูปของคำกริยาเติม -ing หรือคำคำ เดียวกันแต่เขียนแตกต่างกันเมื่อทำหน้าที่ต่างกัน เช่นเป็นคำนาม คำกริยา หรือ คำวิเศษณ์ เป็นต้น เช่น

CONNECT

CONNECTED

CONNECTING

CONNECTION

CONNECTIONS

สเต็มมิ่งอัลกอริทึมช่วยในการแปลงรูปแบบคำต่าง ๆ ให้กลับมาอยู่ในรูปแบบของรากคำเดิม ด้วยการตัด suffix ต่าง ๆ ของคำออกเช่น -ED , -ING , -ION และ -IONS เพื่อให้เหลือรากคำ CONNECT เท่านั้น การทำแบบนี้ช่วยลดจำนวนเทอมที่ถูกจัดเก็บลงในฐานข้อมูลคำซึ่งช่วย ลดขนาดและความซับซ้อนของข้อมูลในฐานข้อมูลด้วย

สเต็มมิ่งอัลกอริทึมที่ได้รับความนิยมและถูกนำไปพัฒนาต่อกันอย่างแพร่หลายคือ Porter อัลกอริทึมซึ่งเกิดขึ้นครั้งแรกในปี 1980 และได้รับความนิยมเป็นอย่างสูง อัลกอริทึมนี้มีมีกฎ การแปลงคำหลายรูปแบบเช่น

- แปลงคำพหูพจน์เป็นเอกพจน์
- การตัด “ing” ออกจากคำ
- การตัด “-ed” และ “-ly” ออก
- การตัด suffix ของคำต่าง ๆ ออกเช่น แปลง "-ational" เป็น "-ate" หรือ แปลง "-alism" เป็น "-al" เป็นต้น

ปัจจุบันอัลกอริทึมนี้สามารถทำงานได้ดีในระดับหนึ่งแต่บางครั้งก็ให้ผลลัพธ์ที่ไม่ถูกต้องนัก กล่าวคือเมื่อแปลงคำแล้วอาจได้คำที่ไม่มีความหมายในภาอังกฤษเช่น คำว่า evolution แปลงเป็น evolute หรือ ปัญหาของการแปลงคำที่เพี้ยนไปเช่น คำว่า engineering แปลงเป็น engine เป็นต้น

อัลกอริทึมนี้ใช้หลักการวิเคราะห์คำทีละตัวอักษรและพิจารณาตัวอักษรข้างเคียงรวมทั้งพิจารณาถึงตัวสระและพยัญชนะด้วย มีกฎในการ จัดการคำที่สั้นเกินไปและคำพิเศษต่าง ๆ พิจารณาการแปลงรูปแบบคำสองคำ W1 และ W2 เพื่อให้ได้รากคำ S ซึ่งสามารถทำได้ ถ้าทั้ง W1 และ W2 มีความหมายเดียวกันภายในประโยคเช่น W1=`CONNECTION' และ W2=`CONNECTIONS' แต่ในกรณีที่ W1=`RELATE' และ W2=`RELATIVITY' โปรแกรมที่ดีไม่ควรแปลง RELATIVITY ให้กลายเป็น RELATE หากในเอกสารที่มีคำว่า RELATIVITY เป็นเอกสารทางด้านฟิสิกส์เพราะคำว่า RELATIVITY จะหมายถึง ทฤษฎีสัมพันธภาพของไอน์สไตน์ การแปลงคำนี้จะทำให้รากคำที่ได้กับเอกสารไม่เกี่ยวข้อง กันเลย การสร้างกฎภายในโปรแกรมจะสามารถใช้กับคำต่าง ๆ ที่เขียนในรูปแบบเดียวกัน ได้เช่นการใช้กฎแปลงคำ SAND และ SANDER โปรแกรมจะใช้กฎเดียวกันแปลงคำ WAND และ WANDER ด้วย ซึ่งในความเป็นจริง -ER ของคำ WANDER ไม่ใช่ suffix ของคำแต่เป็นส่วนหนึ่งของคำ ดังนั้น การทำงานแบบนี้อาจให้ผลลัพธ์ที่ไม่ถูกต้องได้ ดังนั้นโปรแกรมจึงมีการสร้างกฎพิเศษเพื่อช่วยในการพิจารณาคำเหล่านี้และกฎเกณฑ์เพื่อช่วยในการพิจารณาคำที่มีความหมายเดียวกันแต่เขียนในรูปแบบที่ต่างไปจากการเติม suffix เพียงอย่างเดียวเช่น DECEIVE/DECEPTION RESUME/RESUMPTION และ INDEX/INDICES เป็นต้น รายละเอียดของอัลกอริทึมนี้สามารถดูได้ในภาคผนวก ก

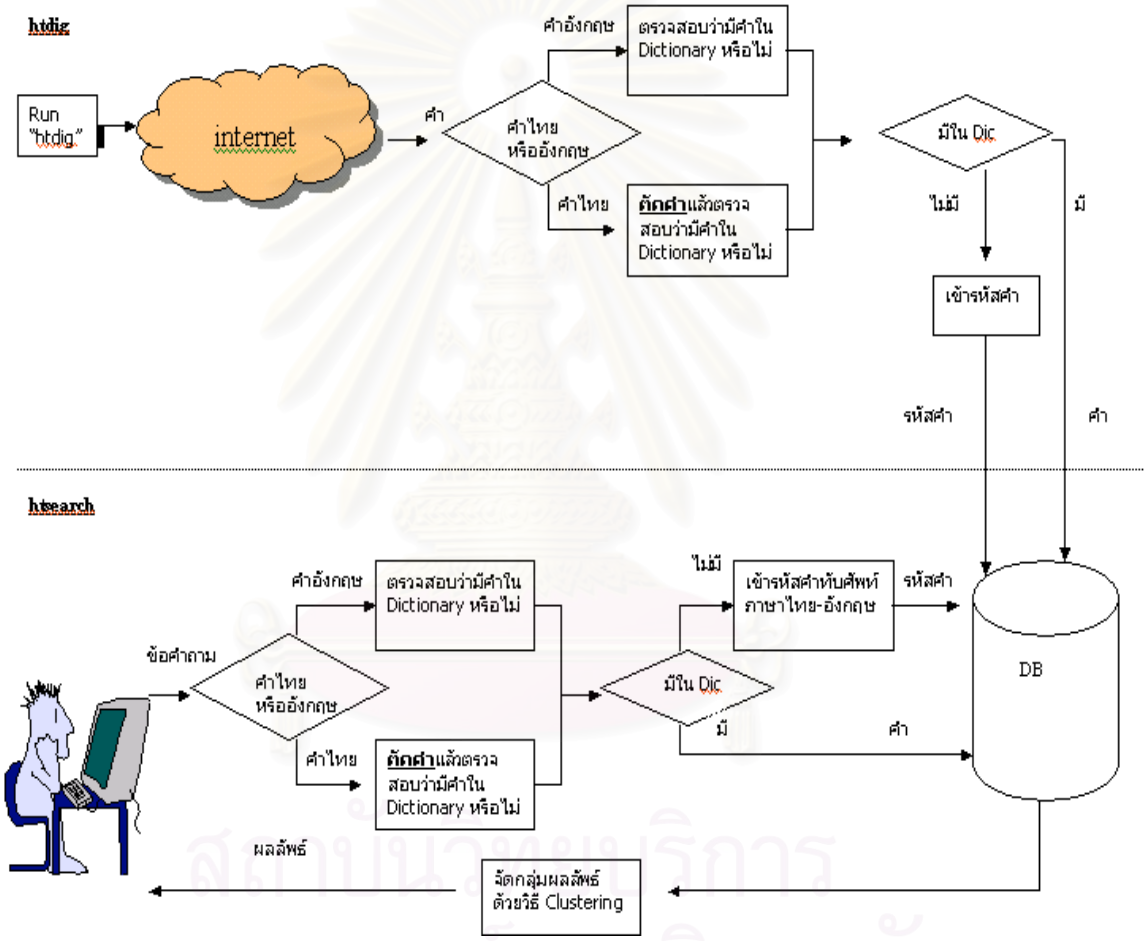
ในงานวิจัยชิ้นนี้จะใช้บางส่วนของสเต็มมิ่งอัลกอริทึมเท่านั้น กล่าวคือจะแปลงคำภาษาอังกฤษในรูปพหูพจน์เป็นเอกพจน์และตัดคำที่มี ing ออก เรียกกระบวนการนี้ว่าไลท์สเต็มมิ่ง อัลกอริทึม (Light Stemming Algorithm) การใช้ไลท์สเต็มมิ่งอัลกอริทึมนี้เนื่องจาก การทำโปรเซสสเต็มมิ่งอัลกอริทึมแบบเต็มรูปแบบทำให้เสียเวลามากและไม่คุ้มค่ากับการทำ

สำหรับภาษาไทยไม่มีอัลกอริทึมการทำ Stemmer เนื่องจากภาษาไทยไม่มีการแปลงรูปคำศัพท์เป็นรูปอื่น ๆ เหมือนคำในภาษาอังกฤษ

บทที่ 3

การวิเคราะห์ออกแบบโปรแกรม

สำหรับงานวิจัยนี้จะพัฒนาระบบค้นคืนสารสนเทศข้ามภาษาไทย-อังกฤษ สำหรับคำทับศัพท์และนำผลลัพธ์ที่ได้มาจัดกลุ่มข้อมูลแบบซัฟฟิกทรี (Suffix Tree Clustering - STC) ซึ่งมีรูปแบบ การทำงานของระบบแสดงได้ดังรูป 3.1



รูปที่ 3.1 ลักษณะการทำงานของระบบค้นคืนในงานวิจัย

รูปที่ 3.1 แสดงการทำงานของระบบค้นคืนในงานวิจัย ซึ่งมีวิธีทำงานตามลำดับขั้นดังนี้

1) การทำดัชนี (Indexing)

ระบบทำการค้นหาข้อมูลเวบเพจต่าง ๆ มาเก็บลงในฐานข้อมูลของตัวเองเพื่อรอการสืบค้นจากผู้ใช้ต่อไปโดยจะสร้างดัชนี (Index) ขั้นตอนนี้จะเกิดขึ้นก็ต่อเมื่อเกิดการเปลี่ยนแปลงแก้ไขข้อมูลในเวบเพจเท่านั้น ระบบจะสร้างดัชนีให้กับเอกสารที่บันทึกอยู่ในแฟ้มข้อมูลประเภท Text และ HTML โดยที่ก่อนการจัดเก็บข้อมูลลงในดัชนี ระบบจะเรียกโปรแกรมตัดคำภาษาไทย เพื่อทำการตัดคำภาษาไทยก่อน และนำคำทั้งภาษาไทยและภาษาอังกฤษไปทำการตรวจสอบ กับคำในพจนานุกรมภาษา หากไม่พบในพจนานุกรมภาษาจะถือว่าคำ ๆ นั้นเป็นคำทับศัพท์ ดังนั้นระบบจะเรียกโปรแกรมแปลงคำทับศัพท์ให้เป็นรหัสคำก่อนจะจัดเก็บเป็นดัชนีคำลงในฐาน ข้อมูลและเก็บคำที่ไม่ใช่คำทับศัพท์ลงในฐานข้อมูลตามปกติ

2) การค้นหา (Searching)

เมื่อผู้ใช้ต้องการค้นหาเอกสารภายในเวบไซต์ จะสามารถกรอกเงื่อนไขการค้นหาผ่าน HTML ฟอร์ม ข้อมูลหรือข้อคำถามที่ระบุจะถูกส่งมาให้โปรแกรมค้นหาเพื่อทำการหาข้อมูล ในดัชนีคำภายในฐานข้อมูลและให้ผลลัพธ์ส่งต่อไปจัดกลุ่มข้อมูลต่อไป การทำงานของ โปรแกรมค้นหาคือจะตรวจสอบว่าหากเป็นคำภาษาไทย ระบบค้นคืนจะเรียกโปรแกรมตัดคำ ภาษาไทยเพื่อทำการตัดคำภาษาไทยก่อนทุกครั้งที่เจอคำภาษาไทย แต่ไม่ทำเช่นนี้กับคำภาษา อังกฤษ จากนั้นจะนำคำสำคัญนั้นไปทำการตรวจสอบกับคำในพจนานุกรมภาษา หากไม่พบ ในพจนานุกรมภาษาจะถือว่าคำ ๆ นั้นเป็นคำทับศัพท์ และระบบจะเรียกโปรแกรม แปลงคำทับศัพท์ให้เป็นรหัสคำก่อนจะนำรหัสคำไปค้นหาเอกสารต่อไป

3) การจัดกลุ่มผลลัพธ์ (Clustering)

ในส่วนนี้ จะทำการรับผลที่ได้จากการค้นหาข้อมูลเพื่อนำเอกสารต่าง ๆ มาจัดกลุ่มข้อมูลตามหลักการการจัดกลุ่มข้อมูลแบบซัฟฟิกทรี (Suffix Tree Clustering) กล่าวคือรายการของเอกสารทั้งหมดจะถูกนำไปจัดการตัดคำทั่วไป (Stop word) ออกและนำคำที่เหลือ ไปผ่านสเต็มมิ่งอัลกอริทึม สุดท้ายรายการเอกสารทั้งหมดจะถูกนำไปสร้างซัฟฟิกทรีเพื่อหา เบสคลัสเตอร์ที่เหมาะสมและจัดกลุ่มเอกสารให้เข้ากันเพื่อแสดงให้ผู้ใช้ต่อไป

บทที่ 4

การพัฒนาโปรแกรม

4.1 การพัฒนาส่วน htdig

เนื่องจากภาษาไทยมีความซับซ้อนมากกว่าภาษาอังกฤษเพราะภาษาไทย 1 ประโยค จะเขียนติดกันจนจบประโยคไม่สามารถแบ่งแยกคำจากกันได้อย่างชัดเจนเหมือนในภาษาอังกฤษ ที่มีช่องว่างระหว่างคำแต่ละคำเสมอ ดังนั้นในการจะที่จะสร้างฐานข้อมูลที่ใช้ในการเก็บคำแต่ละคำโดยดูจากช่องว่างเป็นหลักในการแบ่งแยกคำจึงไม่สามารถทำได้ในทันที ดังนั้นจึงต้องมีการเพิ่มเอาส่วนของการตัดคำภาษาไทยมาใช้ในการแบ่งแยกคำภาษาไทยออกจากกัน โดยการแทรก <wbr> เข้าไประหว่างคำเพื่อที่จะให้ htdig สามารถแบ่งแยกคำภาษาไทยได้และนำมาสร้าง เป็นฐานข้อมูลของคำขึ้นมาได้อย่างถูกต้อง นอกจากนี้เพื่อให้ตรงตามวัตถุประสงค์ของงานวิจัย ให้สามารถค้นคืนข้ามภาษาไทย-อังกฤษสำหรับคำทับศัพท์ จึงต้องมีการเพิ่มส่วนการเข้ารหัสคำสำหรับคำทับศัพท์ก่อนมีการจัดเก็บคำเหล่านี้ลงในฐานข้อมูลด้วย

ขั้นตอนการทำงาน

1) ตั้งค่าเริ่มต้นของการทำงานต่าง ๆ
2) ตรวจสอบค่าเริ่มต้นต่าง ๆ ว่าถูกต้องหรือไม่ เช่น URL ที่ใช้เป็นจุดเริ่มต้นเพื่อทำการเก็บข้อมูลเอกสารต่าง ๆ บนเครือข่ายของระบบ
3) จัดเตรียมฐานข้อมูลที่ใช้ในการจัดเก็บคำ และ URL พร้อมทั้งตัวอย่างรายละเอียดของเอกสารของ URL
4) สร้าง Retriever Object เพื่อใช้ในการเก็บ URL ทั้งหมดที่จะต้องผ่านและเข้าไป เก็บข้อมูลของเอกสารภายในนั้น
5) เริ่มการทำงานเพื่อเก็บข้อมูลคำของเอกสาร โดยเริ่มต้นที่ URL ที่ได้กำหนดไว้ให้เป็น URL เริ่มต้น ก่อนที่จะเริ่มเก็บคำใส่เข้าไปในฐานข้อมูลจะทำการตัดคำไทยก่อน โดยจะมีผลของการตัดคำเฉพาะในส่วนที่เป็นภาษาไทยเท่านั้นโดยจะไม่กระทบต่อคำภาษาอังกฤษรวมทั้ง Tag ในภาษา HTML ก่อนการเก็บข้อมูลจะนำคำมาตรวจสอบกับคำในดิกชันนารีทั้งคำภาษาอังกฤษและคำภาษาไทย หากไม่พบคำนั้นในดิกชันนารีจะถือว่าคำนั้นเป็นคำทับศัพท์ ในกรณีนี้คำนั้นต้องถูกสร้างเป็นรหัสคำก่อนและแปลงเป็นรหัส ASCII เพื่อจัดเก็บลงในฐานข้อมูลต่อไป สำหรับคำทั่วไป ที่ไม่ถือว่าเป็นคำทับศัพท์จะถูกจัดเก็บเป็นคำนั้น ๆ ตามปกติ

6) เมื่อทำการเก็บข้อมูลทั้งหมดภายในขอบเขต URL ที่กำหนดแล้วรวมทั้งสร้าง ฐานข้อมูลที่ใช้ในการเก็บคำเรียบร้อยแล้วก็แสดงผลการทำงานรวมทั้งสถิติต่าง ๆด้วย

การเก็บข้อมูลลงในฐานข้อมูล

ในการเก็บข้อมูลของแต่ละหน้าเว็บเพจมีความจำเป็นที่จะต้องใช้ฐานข้อมูลในการจัดเก็บ ทั้งนี้เพื่อความสะดวกและความรวดเร็วในการสืบค้น ฐานข้อมูลที่ใช้คือ library BerkeleyDB โดยจะแบ่งการจัดเก็บเป็น 2 ส่วนใหญ่ ๆ คือ ส่วน Document Database และส่วน Word Database

Document Database ทำหน้าเก็บข้อมูลต่าง ๆ ของแต่ละเว็บเพจ ซึ่งประกอบไปด้วย

- URL ของเว็บเพจนั้น
- หมายเลขประจำตัวของเว็บเพจ (document id)
- ชื่อเว็บเพจนั้น (title) ตามที่ปรากฏอยู่ใน tag <title>…</title> ของเอกสาร HTML
- สถานะของเว็บเพจนั้น (state) ซึ่งประกอบด้วย 0=ปกติ 1=ไม่พบ 2=ไม่จัดเก็บ index และ 3=เว็บเพจเก่า
- วันที่และเวลาที่เว็บเพจนั้นมีการแก้ไขครั้งล่าสุด (last modification time) ตามที่ได้รับมาจากเว็บเซิร์ฟเวอร์
- ขนาด (size) หน่วยเป็นไบต์
- ข้อความบางส่วนที่อยู่ในหน้าเว็บนั้น (excerpt) เพื่อนำมาใช้ในการแสดงผล
- คำที่อยู่ใน META Description tag ของเอกสาร HTML
- วันที่และเวลาที่เว็บเพจนั้นถูกทำการ index ครั้งสุดท้าย
- จำนวนลิงค์ในเว็บเพจนั้น (outgoing links)
- จำนวนลิงค์ที่ชี้มายังเว็บเพจนั้น (incoming links หรือ backlinks)
- จำนวนลิงค์ที่ห่างจากหน้าเริ่มต้นที่ทำ index (hop count)
- ลายเซ็น (signature) ของเอกสาร เพื่อใช้ในการตรวจสอบความซ้ำซ้อน
- ชื่อของลิงค์ที่ชี้มายังเว็บเพจนั้น (ตัวอย่างเช่น description

Word Database จะทำหน้าที่เก็บคำสำคัญต่าง ๆ ในเว็บเพจซึ่งคำเหล่านั้น จะมีหมายเลขประจำตัวของเว็บเพจ (document id) เพื่อให้สามารถดู URL และข้อมูลต่าง ๆ ได้จาก Document Database นั่นเอง สำหรับคำทับศัพท์จะถูกเก็บเป็นรหัส ASCII ตามด้วยคำต้นแบบ (Original) ของมันเองนั่นคือเก็บในรูปแบบ <รหัสคำ ASCII>|<คำ Original>

4.2 การพัฒนาส่วน htsearch

ดังที่กล่าวมาแล้วในส่วนของการจัดเก็บข้อมูลและการจัดทำดัชนีว่าต้องมีการตัดคำภาษาไทยเพื่อสามารถนำคำที่ตัดแบ่งแล้วไปสร้างเป็นฐานข้อมูลของคำได้ เนื่องจากว่าอัลกอริทึมของการตัดคำภาษาไทยนั้นอาจจะมีปัญหาของการตัดคำได้ไม่ถูกต้องร้อยเปอร์เซ็นต์ ซึ่งบางครั้งอาจจะตัดคำได้ไม่ตรงกับความเป็นจริงที่ควรจะเป็น ซึ่งอาจส่งผลถึงการค้นหาคำๆนั้นไม่เจอเลยก็ได้ หากเรานำคำที่ผู้ใช้ป้อนเข้ามาไปค้นหาในฐานข้อมูลของคำเลยทันที

ขั้นตอนการทำงาน

1) เมื่อผู้ใช้ใส่ข้อมูลที่ต้องการค้นหาในหน้า search form ข้อมูลดังกล่าวจะถูกส่งผ่านเข้ามายังส่วน htsearch เพื่อเป็น input ของการค้นหา
2) เมื่อได้รับ input ที่ส่งผ่านเข้ามาแล้ว จะนำ input ที่เป็น string character มาสร้าง ให้อยู่ในลักษณะที่เป็นรูปแบบเดิมของคำที่ได้รับเข้ามา โดยจะเก็บในรูปแบบของคำ แต่ละคำคั่นด้วยเครื่องหมาย “|” (ลักษณะเป็น Original String Pattern) ทั้งนี้เพื่อประโยชน์ ในการนำมาใช้เทียบกับรายละเอียดบางส่วนของเอกสารผลลัพธ์ ที่ได้จากการค้นหา แล้วทำการเปลี่ยนคำที่ต้องการที่พบในเอกสารนั้น ๆ ให้มีลักษณะ เป็นตัวหนา เพื่อให้ผู้ใช้ สามารถพบคำที่ต้องการหาได้อย่างสะดวก
3) เมื่อทำการสร้าง Original String Pattern แล้ว จะนำ Input เดิม ไปตัดคำภาษาไทยเพื่อแยกกลุ่มคำภาษาไทยให้กระจายออกเป็นคำ ๆ เพื่อให้สามารถนำมาใช้ค้นหาในฐานข้อมูลของคำที่สร้างโดย htdig ได้ โดยเมื่อทำการตัดคำภาษาไทยแล้ว จะนำผลของการตัดคำมาสร้างเป็นรูปแบบ List ของคำที่ต้องการค้นหาทั้งหมด โดยจะประกอบด้วยเครื่องหมายต่าง ๆ ด้วย ได้แก่ เครื่องหมายวงเล็บ , เครื่องหมายในการหาคำแบบเฟส (“) , เครื่องหมายที่เกี่ยวกับ Boolean(+,-,& และ |) รวมทั้งคำเชื่อมต่างๆ ในการค้นหาแบบ Advance Search ได้แก่ AND , OR และ NOT พร้อมทั้งแทรกคำว่า AND ลงไปในช่องว่างระหว่างคำที่ไม่ได้อยู่ภายในเฟสด้วย เพื่อให้มีรูปแบบ (syntax) ที่ถูกต้องโดย node ของ List จะประกอบด้วยคำเพียง คำเดียวหรือเครื่องหมายเพียงเครื่องหมายเดียวเท่านั้น เพราะจะต้องนำไปใช้ใน การตรวจสอบความถูกต้องของ รูปแบบ (syntax) ในขั้นต่อไป
4) นำ List ของคำที่ได้จาก 3 มาตรวจสอบว่าคำใดเป็นคำทับศัพท์โดยนำแต่ละคำ ไปค้นหาในดิชันนารีคำภาษาไทยและภาษาอังกฤษ ถ้าไม่พบในดิกชันนารีคำก็จะถือ ว่าคำนั้นเป็นคำทับศัพท์ ระบบจะแปลงคำนั้นเป็นรหัสคำเพื่อให้ใช้รหัสคำใน การค้นคืน

ข้อมูลแทนคำทับศัพท์คำนั้น แต่คำทั่วไปอื่น ๆ ที่ไม่ถือเป็นคำทับศัพท์ จะยังคงใช้คำนั้น ๆ ในการค้นคืนต่อไป

5) นำ List ของคำที่ได้จาก 4 (SearchWords List) มาตรวจสอบรูปแบบ (syntax) เมื่อทำการตรวจสอบเรียบร้อยแล้ว จะนำผลที่ได้มาสร้างเป็นรูปแบบที่จะสามารถ นำไปแปลความหมายเพื่อให้ได้ผลการค้นหาที่ถูกต้องตามรูปแบบ(Syntax) โดยจะทำ การเปลี่ยนเครื่องหมายต่าง ๆ ให้อยู่ในรูปแบบเดียวกันทั้งหมด ได้แก่
   - เครื่องหมาย “+” , “&” จะถูกเปลี่ยนเป็น AND
   - เครื่องหมาย “|” จะถูกเปลี่ยนเป็น OR
   - เครื่องหมาย “-” จะถูกเปลี่ยนเป็น NOT
6) นำ List ที่เป็นผลมาจากข้อ 5 ไปใช้ในการค้นหาผลลัพธ์ทั้งหมด
7) นำผลลัพธ์ที่ได้จาก 6 ซึ่งเป็นรายการของเอกสารที่ตรงกับข้อคำถาม โดยการเอา คำอธิบายย่อ (excerpt) มาสร้างซัฟฟิกทรีเพื่อทำการจัดกลุ่มของเอกสารและทำการเรียงผลลัพธ์ที่ได้ (URL) ตามลำดับ Score ที่ได้คำนวณขึ้นมา
8) นำ Original Sting Pattern มาเปรียบเทียบกับรายละเอียดของเอกสารทั้งหมด ที่ค้นพบคำที่ต้องการ เพื่อสร้างให้คำที่ต้องการมีลักษณะเป็นตัวอักษรตัวหนาเพื่อ ความสะดวกในการเลือก URL ที่ต้องการใช้งานของผู้ใช้ ระบบค้นคืน
9) แสดงผลการค้นหาทั้งหมดตามรูปแบบที่ได้สร้างไว้

4.3 การพัฒนาส่วน web Interface ของระบบ

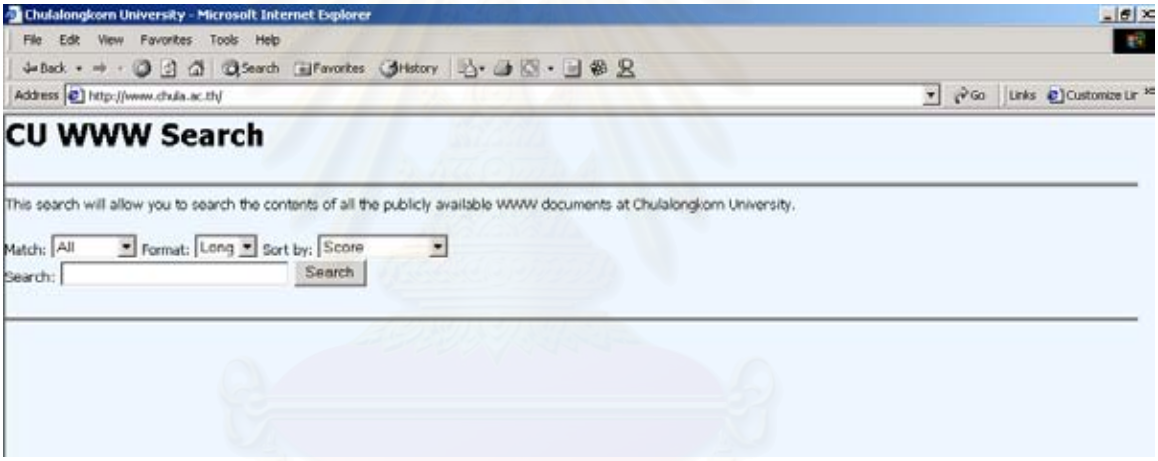
ส่วนนี้เป็นส่วนของการรับข้อมูลจากผู้ใช้เพื่อค้นคืน ซึ่งผู้วิจัยอิงตามรูปแบบของระบบ htdig เดิมแสดงได้ดังรูปที่ 5.2 โดยที่เวบเพจหน้านี้ได้รับการออกแบบฟังก์ชั่นการค้นหา ให้ผู้ใช้สามารถทำการค้นหาได้ดียิ่งขึ้น ดังนี้

1) ส่วน Match มี 3 ฟังก์ชันการทำงานคือ
   - ALL เพื่อการค้นคืนเอกสารที่มีคำทุกคำที่ปรากฏอยู่ในช่องรับข้อคำถาม
   - ANY เพื่อการค้นคืนเอกสารที่มีคำใดคำหนึ่งหรือทุกคำที่ปรากฏอยู่ในช่องรับข้อคำถาม
   - Boolean เพื่อการค้นคืนที่ยินยอมให้ผู้ใช้สามารถใส่ข้อคำถามที่มีคำ AND , OR และ NOT ได้ ช่วยเพิ่มความสามารถในการค้นคืนให้ดียิ่งขึ้น
2) ส่วน Format มีฟังก์ชัน 2 แบบคือ
   - Short เพื่อแสดงเฉพาะ URL ของเอกสารต่าง ๆ เป็นผลลัพธ์
   - Long เพื่อแสดง URL ,คำอธิบายย่อ (snippet) และรายละเอียดของเอกสารต่าง ๆ

เป็นผลลัพธ์

3) ส่วน Sort by ให้ผู้ใช้เลือก เพื่อการเรียงลำดับข้อมูลตามความต้องการดังนี้

- Score เพื่อเรียงลำดับผลลัพธ์ตามคะแนนที่คำนวณได้จากมากไปน้อย
- Time เพื่อเรียงลำดับผลลัพธ์ตามวันที่จากวันล่าสุดไปหาวันที่ก่อนหน้า
- Title เพื่อเรียงลำดับผลลัพธ์ตามลำดับของตัวอักษรของหัวข้อของเอกสารจาก a-z หรือ ก-ฮ
- Reverse Score เพื่อเรียงลำดับผลลัพธ์ตามคะแนนที่คำนวณได้จากน้อยไปมาก
- Reverse Time เพื่อเรียงลำดับผลลัพธ์ตามวันที่จากวันที่ก่อนหน้าไปหาวันที่ล่าสุด
- Reverse Title เพื่อเรียงลำดับผลลัพธ์ตามลำดับของตัวอักษรของหัวข้อของเอกสารจาก z-a หรือ ฮ-ก



รูปที่ 4.1 ส่วนรับข้อมูลเพื่อการค้นคืนของงานวิจัย

บทที่ 5

การทดสอบและแก้ไขโปรแกรม

5.1 เครื่องมือที่ใช้ทดสอบ

วิทยานิพนธ์ชุดนี้ รวมทั้งโปรแกรมต่างๆได้ถูกพัฒนาและได้ทำการทดสอบภายใต้อุปกรณ์ฮาร์ดแวร์และซอฟต์แวร์ดังต่อไปนี้

ฮาร์ดแวร์

1. ระบบทำงานส่วนการค้นคืนข้อมูลเพื่อจัดเก็บลงในฐานข้อมูลของระบบค้นคืน อยู่บนระบบปฏิบัติการ Linux RedHat 7.2 มีหน่วยความจำไม่น้อยกว่า 3 Gb
2. ระบบทำงานส่วน user interface ของระบบค้นคืนอยู่บนระบบปฏิบัติการ Microsoft Windows 2000

ซอฟต์แวร์

1. โปรแกรมระบบปฏิบัติการ Linux RedHat 7.2
2. โปรแกรมบราวเซอร์ Internet Explorer เวอร์ชัน 5.5
3. เครื่องมือการพัฒนาโปรแกรม Microsoft Visual C++ 6.0

ในการทดสอบ ผู้วิจัยได้พัฒนาระบบค้นคืนสารสนเทศข้ามภาษาไทย-อังกฤษสำหรับ คำทับศัพท์และจัดกลุ่มผลลัพธ์ด้วยวิธีการจัดกลุ่มข้อมูลแบบซัฟฟิกทรี ดังนั้นเมื่อมีการทำการทดสอบโดยใช้ระบบค้นคืนของงานวิจัย ผู้วิจัยสามารถทดสอบผลลัพธ์จากการค้นคืนด้วยข้อมูลตัวอย่างต่าง ๆ เปรียบเทียบกับผลลัพธ์ที่ได้จากระบบค้นคืนต้นแบบ htdig (www.htdig.org)

5.2 ข้อมูลที่ใช้ในการทดสอบ

ในการทดสอบนั้น ผู้วิจัยได้ทำการทำคัดเลือกคำทับศัพท์เพื่อใช้เป็นข้อมูลทดสอบต่างๆ ทั้งหมดจำนวน 50 คำ สามารถดูได้ในภาคผนวก ค และแสดงตัวอย่างบางคำในตารางที่ 5.1 ซึ่งแสดงตัวอย่างคำทับศัพท์ทั้งภาษาไทยและภาษาอังกฤษ พร้อมรหัสคำ

ตารางที่ 5.1 ข้อมูลตัวอย่างคำทับศัพท์และรหัสคำ

คำภาษาไทย	รหัสคำ ที่ได้จากการเข้า รหัสคำภาษาไทย	คำภาษาอังกฤษ	รหัสคำ ที่ได้จากการเข้า รหัสคำ ภาษาอังกฤษ
กอบกุล	kbkl$u	Korbkul	kbkl$u
อาทิตย์	ttai	Arthit	ttai
ชูชีพ	ccpUE	Chucheep	ccpXE
เกริก	krkei	Krerk	krkq
เมธี	mte	Matee	mteE
ฐิต	tti	Thit	tti
ทวิตีย์	tvtiE	Twittie	tvti!
วันพร	vnprV	Wunporn	vnpnV$
ชัย	c!	Chai	k!
เชษฐ	cne	Chate	cte
นงลักษณ์	nglkV	Nongluk	nnlk$V
สาธิต	staii	Sartid	stai
ธาราทิพย์	trtpaai	Taratip	rtpaai
วีระ	vrEa	Veera	vrla
อะลูมิเนียม	lmnmaUil	aluminium	lmnmaUil
ออกซิเจน	sjn$ie	oxygen	sjn$ie
อิเล็กตรอน	lktrnie$	electron	lktrnie$
คาร์โบไฮเดรต	kbhdrtao!e	carbohydrate	kbhdrtao!e

5.3 ขั้นตอนการทดสอบ

ผู้วิจัยแบ่งหัวข้อของการทดสอบออกเป็น 3 หัวข้อย่อย ซึ่งมีจุดมุ่งหมายคือเพื่อทดสอบระบบค้นคืนที่จัดสร้างขึ้นและนำผลลัพธ์ที่ได้จากการทดสอบมาคำนวณหรือเปรียบเทียบเพื่อแสดงให้เห็นว่าระบบค้นคืนของงานวิจัยที่สร้างขึ้นมีประสิทธิภาพในการทำงานเปรียบเทียบกับระบบค้นคืนต้นแบบ htdig เป็นอย่างไรโดยมีหัวข้อในการทดสอบต่างๆเช่น การทดสอบเพื่อหา ประสิทธิภาพของการจัดเก็บข้อมูล โดยพิจารณาถึงขนาดและจำนวนข้อมูลที่จัดเก็บบนระบบ ค้นคืนของงานวิจัย การทดสอบความถูกต้องและหาประสิทธิภาพของการค้นคืนเพื่อตรวจสอบ ฟังก์ชันการทำงานของระบบค้นคืนของงานวิจัย พร้อมนำผลลัพธ์การทดสอบที่ได้ไปทำการเปรียบ เทียบเพื่อหาจำนวนผลลัพธ์ที่แตกต่างระหว่างระบบค้นคืนของงานวิจัยกับระบบค้นคืนต้นแบบ htdig และการทดสอบเพื่อหาประสิทธิภาพของการจัดกลุ่มผลลัพธ์

5.3.1. การทดสอบประสิทธิภาพของการจัดเก็บข้อมูล

สำหรับการทดสอบ ทางผู้วิจัยได้นำระบบค้นคืนที่พัฒนาขึ้นและระบบค้นคืนต้นแบบ htdig มาทำการค้นคืนข้อมูลจากฐานข้อมูลภายในคณะวิศวกรรมศาสตร์จุฬาลงกรณ์มหาวิทยาลัย ทดสอบโดยการส่งโปรแกรมโรบอทออกไปค้นคืนข้อมูลจากเวบไซต์ http://www.eng.chula.ac.th มาเก็บลงใน ฐานข้อมูล

หลังจากการทดสอบ ผู้วิจัยได้ผลลัพธ์แสดงรายละเอียดของการจัดเก็บข้อมูล เช่น ขนาดและจำนวนของข้อมูลที่จัดเก็บ เวลาที่ใช้ในการจัดเก็บ และ ความเร็วในการจัดเก็บข้อมูล ซึ่งแสดงดังตารางที่ 5.2

ตารางที่ 5.2 รายละเอียดการจัดเก็บข้อมูลลงฐานข้อมูลของระบบค้นคืน

รายละเอียดการจัดเก็บข้อมูล	ค่าที่ได้จากระบบค้นคืนของงานวิจัย	ค่าที่ได้จากระบบค้นคืนต้นแบบ htdig
Persistent connections	Yes	Yes
HEAD call before GET	Yes	Yes
HTTP Requests	203	203
HTTP KBytes requested	2541	2541
HTTP Average request time	2700 secs	0.22 secs
HTTP Average speed	1.6 Kb/sec	16.563 Kb/sec

จากผลการทดสอบที่ได้ แสดงให้เห็นว่าระบบค้นคืนของงานวิจัยใช้ขนาดฐานข้อมูล ในการจัดเก็บมากกว่าของระบบค้นคืนต้นแบบ htdig เมื่อจัดเก็บจำนวนข้อมูลเท่ากันเนื่องจากระบบค้นคืนของงานวิจัยต้องจัดเก็บส่วนที่เป็นรหัสคำควบคู่กับคำ ในกรณีที่คำนั้นเป็นคำ ทับศัพท์และการตัดคำภาษาไทยซึ่งมีการเพิ่มแทคพิเศษ <wbr> แทรกระหว่างคำทำให้ การจัดเก็บเอกสารภาษาไทยต้องใช้เนื้อที่เพิ่มขึ้น นอกจากนี้ระบบค้นคืนของงานวิจัยก็ใช้เวลา ที่ใช้ในการจัดเก็บข้อมูลมากกว่า เนื่องจากต้องผ่านกระบวนการเข้ารหัสคำก่อนทำการจัดเก็บ ข้อมูลลงในฐานข้อมูล ด้วยเหตุนี้จึงทำให้ความเร็วในการจัดเก็บข้อมูลของระบบค้นคืนของ งานวิจัยช้ากว่าของระบบค้นคืนต้นแบบ htdig อีกด้วย

5.3.2. การทดสอบความถูกต้องและประสิทธิภาพของการค้นคืน

เพื่อทำการทดสอบทางด้านความถูกต้องของการค้นคืนของงานวิจัย ผู้วิจัยได้ทำ การทดลองนำคำศัพท์ทับศัพท์ทั้งหมด 50 คู่ ไปค้นคืนทีละคำศัพท์กับฐานข้อมูล ของระบบค้นคืนของงานวิจัย ด้วยการเปรียบเทียบรหัสคำทับศัพท์แต่ละตัวกับรหัสคำทับศัพท์ ในฐานข้อมูล ผู้วิจัยได้กำหนดให้ระบบค้นคืนของงานวิจัยยินยอมให้รหัสคำที่ได้มาเปรียบเทียบ กันแล้วมีค่าความแตกต่างได้น้อยกว่าหรือเท่ากับ 2 ตำแหน่งเพื่อให้การค้นคืนมีประสิทธิภาพ ที่ดียิ่งขึ้น เพราะรหัสคำที่ได้ในจากคำในภาษาไทยกับภาษาอังกฤษบางคำยังมีค่าความแตกต่าง กันอยู่บ้าง

ตัวอย่าง การทดสอบคำว่า “TECHNOLOGY” และ “เทคโนโลยี” เป็นคำทับศัพท์ที่ตรงกันในภาษาไทย-อังกฤษหรือไม่ จะเริ่มต้นจากการเข้ารหัสคำแล้วคำนวณหา ค่าความแตกต่างตามแนวทางงานวิจัยของ ทัศวรรณ ศูนย์กลาง [10] ซึ่งใช้การเปรียบเทียบรหัสคำ แบบประมาณ (Approximate Matching) โดยในงานวิจัยนี้กำหนดให้ค่าความแตกต่างมีค่า น้อยกว่าหรือเท่ากับ 2 ตำแหน่ง จะสรุปว่าเป็นคำทับศัพท์ที่ตรงกันในภาษาไทย-อังกฤษ

<u>การเข้ารหัสคำ</u>

TECHNOLOGY → tknlyeooE

เทคโนโลยี → tknljeooE

BOONSERM → Bnsrmuei

บุญเสริม → bnsm$l

การคำนวณหาค่าความแตกต่าง

ตัวอย่าง

1. ค้นหาคำว่า Technology

Edit (tknlyeooE, tknljeooE) = 1

จากตัวอย่างที่ 1 รหัสคำทั้งสองมีค่าความแตกต่างเป็น 1 ดังนั้นก็จะสามารถค้นคืน “TECHNOLOGY” จาก “เทคโนโลยี” ได้ หรือสามารถค้นคืน “เทคโนโลยี” จาก “TECHNOLOGY” ได้เช่นกัน

2. ค้นหาคำว่า Boonserm

Edit (Bnsrmuei, bnsm$I) = 5

จากตัวอย่างที่ 2 รหัสคำทั้งสองมีค่าความแตกต่างเป็น 5 ดังนั้นก็จะไม่สามารถค้นคืน “BOONSERM” จาก “บุญเสริม” ได้ และ ไม่สามารถค้นคืน “บุญเสริม” จาก “BOONSERM” ได้เช่นกัน

สำหรับการทดสอบประสิทธิภาพการค้นคืนสามารถทำได้โดยใช้เวบบราวเซอร์ Internet Explorer เวอร์ชั่น 5.5 ทำการค้นหาข้อมูลตัวอย่างจากตารางที่ 5.1 จากฐานข้อมูล ของระบบค้นคืนของงานวิจัย และทำการค้นหาข้อมูลตัวอย่างเดียวกันจากฐานข้อมูลของระบบ ค้นคืนต้นแบบ htdig แสดงผลเปรียบเทียบจำนวนผลลัพธ์ที่ได้จากการค้นคืนเทียบกันทีละ ตัวอย่างแสดงได้ดังตารางที่ 5.3

ตารางที่ 5.3 ตารางแสดงผลลัพธ์การทดสอบความถูกต้องของโปรแกรม

คำที่ใช้ในการทดลอง		จำนวนเวบเพจที่ได้จากการค้นคืนด้วยระบบค้นคืนของงานวิจัย			จำนวนเวบเพจที่ได้จากการค้นคืนด้วยระบบค้นคืนต้นแบบ htdig แบบ Boolean (เช่น ซิลเวอร์ หรือ Silver)
ภาษาไทย	ภาษาอังกฤษ	ภาษาไทย	ภาษาอังกฤษ	ค้นหาแบบ Boolean (เช่น ซิลเวอร์ หรือ Silver)	
กอบกุล	Korbkul	4	4	4	4
อาทิตย์	Arthit	15	15	15	7
ชูชีพ	Chucheep	4	4	4	4
เกริก	Krerk	3	3	3	3
เมธี	Matee	2	2	2	2
ฐิต	Thit	15	15	15	8
ทวิตีย์	Twittie	12	12	12	11
วันพร	Wunporn	6	6	6	5
ชัย	Chai	23	23	23	23
เชษฐ	Chate	12	12	12	8
นงลักษณ์	Nongluk	14	14	14	11
สาธิต	Sartid	10	10	10	8
ธาราทิพย์	Taratip	9	9	9	9
วีระ	Veera	10	10	10	9
อะลูมิเนียม	aluminium	13	13	13	7
ออกซิเจน	oxygen	11	11	11	9
อิเล็กตรอน	electron	11	11	11	8
คาร์โบไฮเดรต	carbohydrate	12	12	12	7

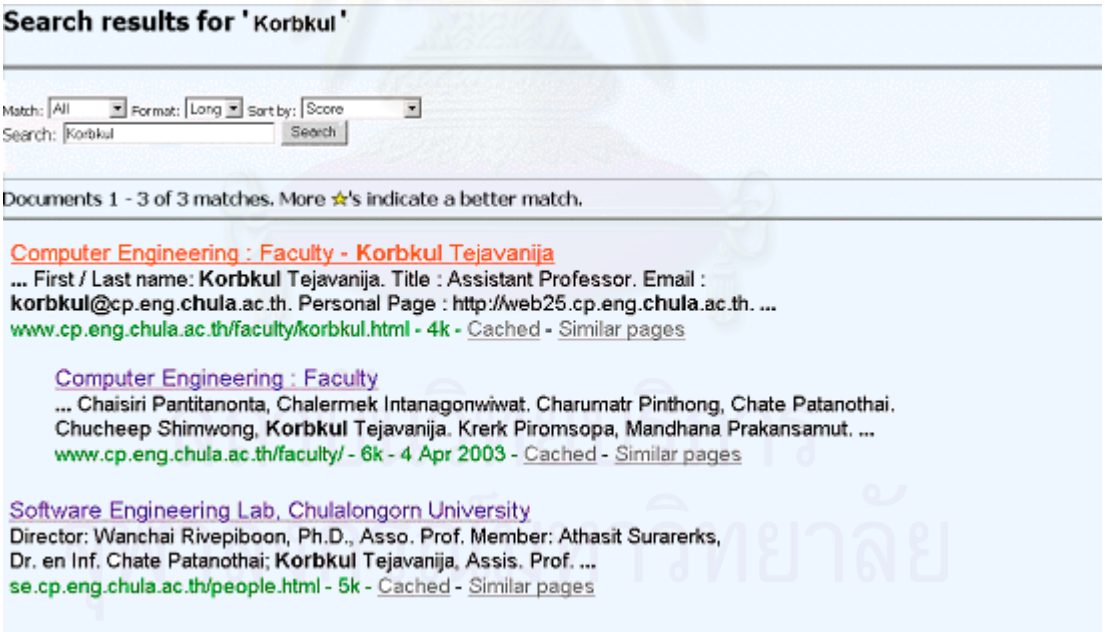
จากการทดลองพบว่าจำนวนผลลัพธ์ที่ได้จากการค้นคืนตัวอย่างในการทดลองโดยระบบค้นคืนของงานวิจัยมีค่ามากกว่าการค้นคืนโดยระบบค้นคืนต้นแบบ htdig และมีค่าน้อยกว่าในบางกรณี การเข้ารหัสคำในการทดลองให้ผลที่ใกล้เคียงกับการค้นคืนต้นแบบ htdig แต่สามารถค้นคืนได้ด้วยคำใดคำหนึ่งในภาษาอังกฤษหรือภาษาไทย โดยที่ไม่จำเป็นต้องทำการค้นคืนแบบ Boolean หรือค้นคืนสองครั้งเนื่องจากการใช้เทคนิคการเข้ารหัสคำที่ดีเข้าช่วย นอกจากนี้การผสม

ผสานกับการตัดคำภาษาไทยที่มีประสิทธิภาพก็ช่วยก็สามารถทำให้การค้นคืนภาษาไทยมี ประสิทธิภาพยิ่งขึ้นด้วย

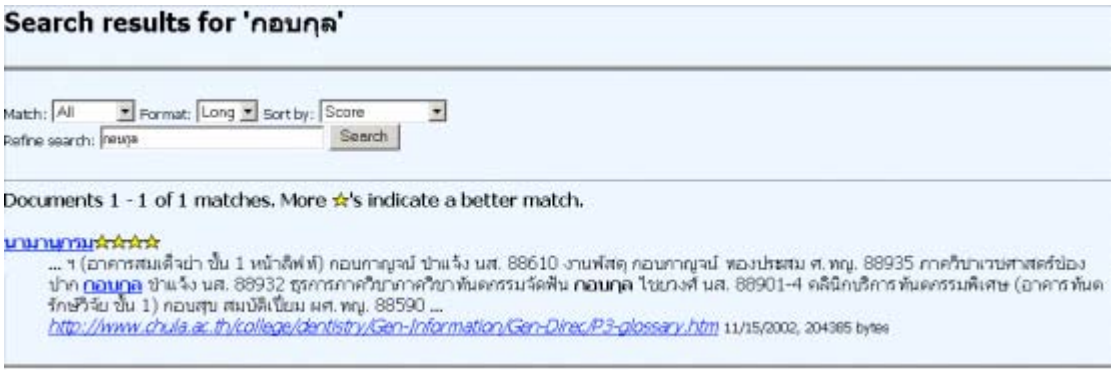
5.3.3 การทดสอบเพื่อหาประสิทธิภาพของการจัดกลุ่มผลลัพธ์

ในหัวข้อการทดสอบนี้ ทางผู้วิจัยได้นำผลลัพธ์ที่ได้จากการค้นคืนและถูกจัดกลุ่มแล้วด้วยวิธีแบบซัฟฟิกทรี ซึ่งนำผลลัพธ์ที่เป็นเอกสารภาษาไทยและภาษาอังกฤษมาทำการจัดแบ่งเอกสารที่เหมือนกันมาอยู่ในกลุ่มเดียวกัน และแยกเอกสารที่ต่างกันออกไปอยู่ในกลุ่มอื่น

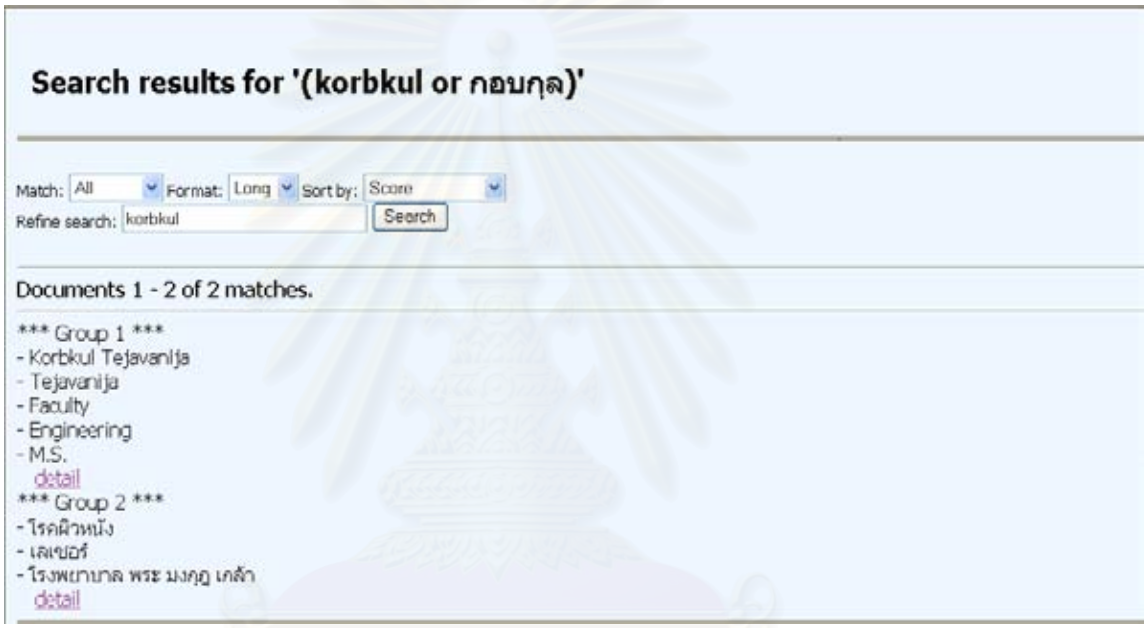
จากรูปที่ 5.1 และ 5.2 แสดงลักษณะของผลลัพธ์ที่ได้จากการค้นคืนคำว่า “กอบกุล” ด้วยระบบค้นคืนภายในต้นแบบ htdig ที่ไม่มีการจัดกลุ่มข้อมูลเลย และลักษณะของ ผลลัพธ์ที่ได้จากการค้นคืนด้วยระบบค้นคืนของงานวิจัยที่จัดกลุ่มผลลัพธ์ใหม่ให้เป็นกลุ่ม ซึ่งใน การทดสอบเพื่อแสดงให้เห็นว่าระบบค้นคืนของงานวิจัยช่วยจัดกลุ่มข้อมูลอย่างชัดเจนเพื่อให้ผู้ใช้ สามารถเลือกหัวข้อที่คิดว่าตรงกับความสนใจก่อนเข้าไปเลือกดูเอกสารที่ตรงกับความต้องการจริง ต่อไป



รูปที่ 5.1 ลักษณะการแสดงผลลัพธ์จากการค้นคืนคำว่า “Korbkul” ของระบบค้นคืนต้นแบบ htdig



รูปที่ 5.2 ลักษณะการแสดงผลลัพธ์จากการค้นคืนคำว่า “กอบกุล” ของระบบค้นคืนต้นแบบ htdig



รูปที่ 5.3 ลักษณะการแสดงผลลัพธ์จากการค้นคืนคำว่า “กอบกุล” ของระบบค้นคืนในงานวิจัย

การทดสอบเริ่มจากผู้วิจัย ได้ทำการสำรวจเอกสารภายในฐานข้อมูลทั้งหมดที่เกี่ยวกับคำตัวอย่างเพื่อดูจำนวนรวมของเอกสารทั้งหมด แล้วทำการค้นคืนคำตัวอย่างนั้นในฐานข้อมูล จากนั้นตรวจสอบกลุ่มผลลัพธ์ที่ได้ และจำนวนเอกสารภายในกลุ่มแต่ละกลุ่มเพื่อนำมาทำการคำนวณ หาค่าความแม่นยำในการจัดกลุ่มข้อมูลเอกสารของระบบค้นคืนด้วยสูตร

$$\text{ค่าความแม่นยำ} = \frac{\sum_{k=1}^{K} Rel_k}{\sum_{\kappa=1}^{K}\left(Rel_k + Non_k\right)} \quad X \quad 100$$

Rel_k หมายถึง จำนวนของเอกสารที่เกี่ยวข้องกับเอกสารส่วนใหญ่ที่คืนกลับมาภายใน

คลัสเตอร์ k

Non_k หมายถึง จำนวนของเอกสารที่ไม่เกี่ยวข้องกับเอกสารส่วนใหญ่ที่คืนกลับมาภายใน

คลัสเตอร์ k

อธิบายได้ว่า เราสามารถวัดค่าความแม่นยำของการจัดกลุ่มข้อมูลได้จากสัดส่วนของผลรวมทั้งหมดของจำนวนเอกสารที่มีเนื้อหาเดียวกันกับเอกสารส่วนใหญ่ในแต่ละกลุ่มย่อยและผลรวมของจำนวนเอกสารทั้งหมดที่คืนกลับมาในแต่ละกลุ่ม

ตารางที่ 5.4 จำนวนเอกสารภายใต้หัวข้อการจัดกลุ่มจากการค้นคืนตัวอย่างคำในการทดลอง

ตัวอย่างข้อมูลที่ใช้ทดลอง	**จำนวนกลุ่มผลลัพธ์**	$\sum Rel_k$	$\sum Non_k$	ค่าความแม่นยำในการจัดกลุ่มข้อมูล
กอบกุล/Korbkul	2	4	0	100
อาทิตย์/Arthit	4	9	6	60
ชูชีพ/Chucheep	2	4	0	100
เกริก/Krerk	1	3	0	100
เมธี/Matee	1	2	0	100
ฐิต/Thit	5	8	7	53.33
ทวิตีย์/Twittie	5	8	5	61.53
วันพร/Wanporn	4	5	1	83.33
ชัย/Chai	11	19	7	73.07
เชษฐ/Chate	5	10	5	66.66
นงลักษณ์/Nongluk	4	10	4	71.43
สาธิต/Sartid	7	14	4	77.77
ธาราทิพย์/Taratip	3	9	2	81.81
วีระ/Veera	7	9	4	69.23
อะลูมิเนียม/Alumineum	7	13	3	81.25
ออกซิเจน/Oxygen	6	12	4	75
อิเล็กตรอน/Electron	6	14	3	82.35
คาร์โบไฮเดรต/Carbohydrate	5	12	2	85.71

จากตาราง 5.4 แสดงค่าความแม่นยำในการจัดกลุ่มข้อมูลของงานวิจัย ซึ่งภายหลังการคำนวณรวมแล้วพบว่าจะมีค่าแม่นยำเฉลี่ยสูงถึง 79.03 เปอร์เซ็นต์

5.4 สรุปผลลัพธ์ของการใช้ระบบค้นคืนข้ามภาษาไทย-อังกฤษ สำหรับคำทับศัพท์และจัดกลุ่มผลลัพธ์ด้วยวิธีซัฟฟิกทรีของงานวิจัย

1. จากการทดลองพบว่าการยินยอมให้การเปรียบเทียบรหัสคำมีค่าความแตกต่างน้อยกว่าหรือเท่ากับ 2 แล้วถือว่าคำทั้งสองเป็นคำเดียวกันนั้น ให้ผลลัพธ์ที่ดีกับคำที่มีความยาวมากเช่น คำว่า ออกซิเจนหรือ คาร์โบไฮเดรต และได้ผลลัพธ์ที่ไม่ดีนักกับคำที่มีความยาวน้อยเช่น คำว่า Thit หรือ ฐิต เนื่องจากคำที่มีความยาวน้อยจะได้รหัสคำที่น้อยและมีโอกาสที่จะไปมีเสียงคล้ายคลึงกับคำอื่นได้มากขึ้น ดังนั้นการพิจารณาเลือกค่า k ที่เหมาะสมจะทำให้ประสิทธิภาพการค้นคืนดียิ่งขึ้นได้

2. คำที่มีความยาวสั้นมักมีเสียงคล้ายคลึงกันมาก จึงทำให้ได้รหัสคำที่ใกล้เคียงกับคำอื่นมากขึ้น ทำให้การค้นคืนได้ผลลัพธ์เป็นเอกสารที่ไม่ตรงกับคำหรือหัวข้อมากขึ้น

3. คำอธิบายย่อ (Snippet) ที่ได้จากบางเวบไซต์ที่ให้รายละเอียดในการอธิบายลักษณะของเวบนั้น ๆ ได้ไม่ดีนัก จะมีทำให้ผลการจัดกลุ่มเพี้ยนไปได้

บทที่ 6

สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

6.1 สรุปผลการวิจัย

เนื่องจากปัจจุบันเครือข่ายอินเตอร์เน็ต เป็นแหล่งข้อมูลสำคัญที่มีประโยชน์ในการศึกษาค้นคว้าหาความรู้เป็นแหล่งกระจายข่าวและสื่อสารข้อมูลในด้านต่าง ๆ ที่นับวันจะมีปริมาณข้อมูลมากขึ้นทุกวัน การที่ต้องเข้าไปค้นหาข้อมูลที่ต้องการบนโลกของอินเตอร์เน็ตจึงเปรียบได้กับการเดินเข้าไปในห้องสมุดขนาดใหญ่เพื่อหาหนังสือหรือข้อมูลจากแหล่งข้อมูลต่าง ๆ ให้ได้ตรงกับความต้องการให้ได้ดีที่สุดแต่จะทำได้อย่างไรถ้าไม่มีเครื่องมือช่วยในการค้นหาเลย การไล่หาข้อมูลด้วยตนเองคงเป็นการเสียเวลามาก หรือหากพบสิ่งที่ต้องการแล้วจะแน่ใจได้อย่างไรว่ายังมีข้อมูลอื่น ๆ ที่ดีหรือเหมาะสมกับสิ่งที่ต้องการมากกว่าหรือไม่ นี่จึงเป็นจุดเริ่มต้นในการพัฒนา งานวิจัยนี้ที่ได้ทำการศึกษาการทำงานของระบบค้นคืนที่มีอยู่ในปัจจุบัน และพยายามที่จะเพิ่ม ประสิทธิภาพของการค้นคืนให้ดียิ่งขึ้น โดยการศึกษาและพัฒนาส่วนที่เกี่ยวกับการค้นคืนคำไทย และการตัดคำภาษาไทยเพื่อการค้นคืนข้ามภาษา การเข้ารหัสคำเพื่อการค้นคืนคำทับศัพท์ที่ช่วย เพิ่มประสิทธิภาพการค้นคืนข้ามภาษาได้ดียิ่งขึ้น

นอกจากนี้งานวิจัยนี้ยังได้มุ่งพัฒนาส่วนแสดงผลลัพธ์ของการค้นคืนซึ่งเป็นส่วนสำคัญที่มีผลต่อความพึงพอใจของผู้ใช้ระบบ โดยมีวัตถุประสงค์ให้ระบบค้นคืนของงานวิจัยสามารถ จัดแบ่งกลุ่มของเอกสารเพื่อช่วยให้การค้นคืนค้นหาเวบไซต์ได้ง่ายและสะดวกยิ่งขึ้น

จากผลการทดลองจะเห็นว่าการค้นคืนด้วยระบบค้นคืนที่มีอยู่ในปัจจุบันยังไม่สนับสนุนการค้นคืนข้ามภาษาไทย-อังกฤษสำหรับคำทับศัพท์ การใส่ข้อคำถามที่เป็นภาษาใดภาษาหนึ่ง จะค้นคืนผลลัพธ์ในภาษานั้นมาเป็นผลลัพธ์ให้เท่านั้น ซึ่งหากผู้ใช้ต้องการค้นคืนทั้งสองภาษา ต้องทำการค้นคืนสองครั้งแล้วพิจารณาผลลัพธ์ที่ได้ในแต่ละครั้งเอาเอง นอกจากนี้การค้นคืน ส่วนใหญ่จะจัดอันดับของผลลัพธ์เป็นรายการเอกสารเรียงตามลำดับคะแนน ผู้ใช้ต้องเลือกดู เอกสารที่ตรงกับความต้องการจริงซึ่งบางครั้งรายการเอกสารที่ยาวมาก ๆ ก็ทำให้ผู้ใช้ต้องเสียเวลา หาสิ่งที่ต้องการพอสมควร จากการทดลองค้นคืนโดยงานวิจัยช่วยลดเวลาในการค้นคืนคำ ทับศัพท์และให้ผลลัพธ์ที่ใกล้เคียงกับการค้นคืนแบบการค้นคืนทีละภาษาด้วยระบบค้นคืนทั่วไป การจัดกลุ่มของเอกสารสามารถแสดงหัวข้อที่เกี่ยวข้องกับเอกสารภายในกลุ่มได้ดีในระดับหนึ่ง

6.2 ปัญหาที่ได้รับ

- เว็บไซท์บางเว็บสร้างขึ้นโดยใช้โปรแกรมสำเร็จรูป ทำให้ส่งข้อมูลที่เป็นตัวอักษรภาษาไทยผิดพลาดนอกจากนั้นเว็บส่วนใหญ่ไม่ได้สร้างมาเพื่อรองรับระบบค้นคืนกล่าวคือ ไม่มี META tag ที่จะช่วยในการค้นหา เช่น ข้อคำถาม (Keywords) หรือ รายละเอียด (Description) เป็นต้น ทำให้การค้นหามีความแม่นยำลดน้อยลง
- การตรวจสอบว่าส่วนจัดทำดัชนี ได้เก็บข้อมูลได้ครบถ้วนทุกเว็บไซท์แล้วทำได้ยากเพราะไม่สามารถทราบจำนวนเว็บเซิร์ฟเวอร์ทั้งหมดได้จึงทำได้เพียงการประมาณเท่านั้น
- โปรแกรม ht://Dig ที่ใช้เป็นต้นแบบในการพัฒนาระบบ มีเอกสารประกอบน้อยมากทำให้มีความยากลำบากในการทำงานเพราะต้องไล่ซอร์สโคด (source code) ทั้งหมด
- การจัดเก็บดัชนีใช้เวลานาน ดังนั้นในการใช้งานจริงต้องคำนึง downtime ของเซิร์ฟเวอร์ด้วยเพราะในขณะที่ทำการเก็บดัชนีอยู่นั้นจะไม่สามารถทำการสืบค้นข้อมูล ได้เลย วิธีแก้ไขอาจทำโดยให้ htdig ไปสร้างฐานข้อมูลใหม่อีกที่หนึ่งแล้วจึงย้ายฐานข้อมูล ใหม่มาทับของเดิมแต่วิธีนี้ต้องการพื้นที่มาก (อย่างน้อย 2 เท่าของขนาดฐานข้อมูล) ด้วยขนาดของฐานข้อมูล ดัชนีที่ใหญ่ทำให้ต้องใช้พื้นที่ (disk space)และหน่วยความจำ (RAM) มากในการจัดเก็บ และประมวลผล
- การจัดเก็บดัชนีจากแหล่งข้อมูลปริมาณมากจะทำให้ระบบเกิดปัญหา Segmentation Fault เนื่องจากฟังก์ชันการทำงานของโปรแกรม ExternalParser.cc ซึ่งทำการบีบอัด ข้อมูล (compression) หรือจัดทำดัชนี (indexing) กับอักขระพิเศษ (Accented Character) ที่ไม่ใช่อักขระมาตรฐาน (Base Character)
- การจัดกลุ่มข้อมูลแบบซัฟฟิกทรี ยังให้ผลลัพธ์ที่ไม่ดีนัก เนื่องจากวิธีนี้ต้องผ่านการทำสเต็มมิ่ง (Stemming) คือ การเปลี่ยนแปลงคำต่าง ๆ เป็นรากศัพท์ ก่อนการสร้างต้นไม้ของคำซึ่งโปรแกรมสเต็มมิ่งในปัจจุบันยังทำงานได้ไม่ดีเช่น แปลงคำว่า engineering เป็น engin (ควรแปลงเป็น engineer) หรือ แปลงคำว่า evolution เป็น evolut (ซึ่งไม่ปรากฏคำนี้ว่ามีในดิกชันนารีคำภาษาอังกฤษในปัจจุบัน) เป็นต้น การแปลงคำที่ ไม่ถูกต้องเหล่านี้ส่งผลให้ระบบค้นคืนของงานวิจัยเข้าใจผิดคิดว่าเป็นคำทับศัพท์ ระบบจึง ทำการเข้ารหัสคำเพื่อการค้นคืนเป็นการทำงานที่ไม่ถูกต้อง
- การจัดกลุ่มข้อมูลแบบซัฟฟิกทรี จะนำเอารายละเอียดย่อ (snippet) ของเอกสารมาสร้างเป็นต้นไม้ของคำ ซึ่งรายละเอียดย่อยนั้นเป็นกลุ่มคำขนาดเล็กทำให้การประเมินความสำคัญของคำหรือการให้น้ำหนักแต่ละคำของเอกสารนั้นได้ผลลัพธ์ที่ไม่ถูกต้องเท่าที่ควร

- การประเมินความสำคัญของคำหรือการให้นำหนักแต่ละคำที่ผิดเพี้ยนไปส่งผลให้การแสดง หัวข้อของการจัดกลุ่มข้อมูลสื่อความหมายได้ไม่ดีและชัดเจน
- ปัจจุบันยังไม่มีโปรแกรมหรือหลักการสเต็มมิ่งภาษาไทยอย่างชัดเจน เพื่อช่วยเพิ่มประสิทธิภาพการค้นคืนภาษาไทยให้ดียิ่งขึ้น

6.3 ข้อเสนอแนะ

ผู้วิจัยพบว่ามีข้อเสนอแนะบางประการที่น่าจะเป็นประโยชน์ และสามารถนำไปใช้ในการพัฒนาระบบค้นคืนข้ามภาษาไทย-อังกฤษสำหรับคำทับศัพท์เพื่อให้ได้ระบบค้นคืนที่มี ประสิทธิภาพที่ดีขึ้นดังนี้

- เนื่องจากงานวิจัยชิ้นนี้ถูกพัฒนาขึ้นเพื่อทำการค้นคืนคำทับศัพท์สำหรับคำภาษาไทยและภาษาอังกฤษ ซึ่งยังคงมีปัญหาในการจัดการกับคำภาษาไทยโดยยังไม่สามารถระบุได้ชัดเจนว่าคำใดในเอกสารเป็นคำทับศัพท์หรือคำนามเฉพาะ และในงานวิจัยชิ้นนี้สามารถระบุคำนามเฉพาะที่มีคำนำหน้าชื่อเช่น อ. หรือ รศ. เป็นต้น และกับคำทับศัพท์อื่น ๆ ที่ผ่านการตัดคำภาษาไทยมาได้อย่าง ถูกต้องเท่านั้น ดังนั้นการศึกษาส่วนการระบุว่าคำใดในภาษาไทยที่เป็นคำ ทับศัพท์ได้อย่างชัดเจนขึ้นจะช่วยให้การพัฒนาระบบค้นคืนที่เกี่ยวกับภาษาไทย ทำงานได้อย่างมีประสิทธิภาพดีขึ้นได้ด้วย

รายการอ้างอิง

[1] David Filo, Jerry Yang. Stanford University in California, Available From : URL http://www.yahoo.com

[2] Available From : URL http://www.altravista.com

[3] Available From : URL http://www.google.com

[4] Available From : URL http://www.excite.com

[5] Dawid Weiss. A Clustering Interface for Web Search Results in Polish and English. : Poznañ University of Technology,Poland , 2001.

[6] Oren Zamir, Oren Etzioni. Grouper: A Dynamic Clustering Interface to Web Search Results. Computer Networks Amsterdam, Netherlands ,Available From : URL http://citeseer.nj.nec.com, 1999.

[7] Oren Zami,Oren Etzioni . Web Document Clustering. Web Document Clustering : A Feasibility Demonstration. Department of Computer Science and Engineering University of Washington , 1998.

[8] Available From : URL http://www.vivismo.com

[9] ประยุทธ สุวรรณวิสารท. การเข้ารหัสคำทับศัพท์เพื่อการค้นคืนข้ามภาษาไทย-อังกฤษ. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย,2541.

[10] ทัศนวรรณ ศูนย์กลาง. การเข้ารหัสคำทับศัพท์ภาษาไทย/อังกฤษเพื่อการค้นคืน ข้ามภาษาด้วยเทคนิคนิวรอลเน็ตเวิร์ก. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย,2541.

[11] UKKONEN, E. On-line Construction of Suffix-Trees. Algorithmica, 14:249-260, 1995.

[12] Available From : URL http://www.tartarus.org/~martin/PorterStemmer

ภาคผนวก

ภาคผนวก ก

สเต็มมิ่งอัลกอริทึม (Stemming algorithm)

การใช้สเต็มมิ่งอัลกอริทึมเพื่อลดรูปคำในภาษาอังกฤษให้เป็นรากศัพท์ วิธีนี้มีประโยชน์สำหรับการทำงานของระบบ IR (Information Retrieval) มากเนื่องจากช่วยในการจัดการข้อมูลคำเดียวกันแต่อยู่ในรูปแบบที่แตกต่างกันให้สามารถสืบค้นได้เหมือนกัน และลดปริมาณข้อมูลที่จะจัดเก็บบนฐานข้อมูลด้วย เพราะเก็บคำต่าง ๆ ในรูปของรากศัพท์ ดังนั้นคำที่เขียนต่างกันจะถูกเก็บเป็นคำเดียวและมีลิงค์ไปยังเอกสารต่าง ๆ กันได้ ภายในอัลกอริทึมมีหลักการทำงานและหลักในการนิยามพยัญชนะและสระ ในภาษาอังกฤษดังนี้

<u>พยัญชนะภาษาอังกฤษ</u> หมายถึงตัวอักษรภาษาอังกฤษใด ๆ ที่ไม่ใช่ A, E, I, O หรือ U และ

Y ที่ไม่ได้ทำหน้าที่เป็นสระภายในคำเช่น คำ TOY มีพยัญชนะคือ T และ Y แต่คำว่า SYZYGY มีพยัญชนะคือ S, Z และG.

<u>สระภาษาอังกฤษ</u> หมายถึง อักษรใด ๆ ที่ไม่ใช่พยัญชนะ

แสดงพยัญชนะด้วย c

แทนสระด้วย v

แทนชุดพยัญชนะ ccc... ซึ่งมีความยาวมากกว่า 0 ด้วย C

แทนชุดสระ vvv... ซึ่งมีความยาวมากกว่า 0 ด้วย V

ดังนั้นคำใด ๆ หรือส่วนของคำใด ๆ สามารถเขียนให้อยู่ในรูปใดรูปหนึ่งต่อไปนี้

CVCV ... C

CVCV ... V

VCVC ... C

VCVC ... V

ซึ่งสามารถเขียนให้อยู่ในรูปเดียวกันดังนนี้

[C]VCVC ... [V]

ตัวอักษรใน [] แสดงถึงมีชุดอักษร (C หรือ V โดยที่ ตัว C หมายถึงชุดพยัญชนะ และ ตัว V หมายถึงชุดสระ) ตั้งแต่ 1 ชุดขึ้นไปหรือไม่มีเลย

ให้สัญลักษณ์ (VC){m} แสดงการมี VC ซ้ำกัน m ครั้ง ดังนั้นสามารถแปลงรูปแบบของคำใหม่ได้ดังนี้

[C](VC){m}[V].

m แสดงจำนวนครั้งที่ชุดอักษรที่มีชุดสระ V ตามด้วยชุดพยัญชนะ C ปรากฏในคำ ถ้า m = 0 หมายถึงไม่มีชุดอักษร VC ปรากฏในคำเลย เช่น

m=0 TR, EE, TREE, Y, BY.

m=1 TROUBLE, OATS, TREES, IVY.

m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

กฎในการแปลงรูปแบบคำ สามารถเขียนได้ในรูป

(condition) S1 -> S2

หมายถึงถ้าคำ ๆ หนึ่งจบด้วย suffix S1 และ ชุดอักษรก่อนหน้า S1 เข้าข่ายเงื่อนไข ที่กำหนดใน condition แล้ว โปรแกรมจะแทนที่ S1 ด้วย S2 เงื่อนไข condition มักถูกกำหนด ในรูปแบบของ m ดังนี้

(m > 1) EMENT ->

ในตัวอย่าง S1 คือ `EMENT' และ S2 ไม่มีคำ ดังนั้นโปรแกรมจะแปลงคำว่า REPLACEMENT เป็น REPLAC เพราะ REPLAC เป็นส่วนหนึ่งของคำโดยมี m = 2 (มี VC 2 ชุด คือ “EPL” และ “AC”)

ส่วนของ condition อาจถูกกำหนดในรูปแบบอื่น ๆ ได้เช่น

*S หมายถึง คำจบด้วย S

v หมายถึง คำที่มีสระปรากฏอยู่

*d หมายถึง คำที่จบด้วยพยัญชนะ 2 ตัวเช่น -TT และ -SS

*o หมายถึง คำที่จบด้วย cvc(ตัวพยัญชนะตามด้วยสระและพยัญชนะ) ซึ่ง c ไม่ใช่ W, X หรือ Y เช่น -WIL และ -HOP

และในส่วนเงื่อนไขสามารถมีได้มากกว่า 1 เงื่อนไขเชื่อมกันด้วย and , or และ not ดังนี้

(m>1 and (*S or *T)) แสดงเงื่อนไขของคำที่มี VC มากกว่า 1 และจบด้วย S หรือ T

(*d and not (*L or *S or *Z)) แสดงเงื่อนไขของคำที่จบด้วยพยัญชนะ 2 ตัวที่ไม่ใช่ L,S หรือ Z

คำต่าง ๆ จะถูกประมวลผลด้วยกฎข้อใดข้อหนึ่งเท่านั้น ด้วยการพิจารณาหา S1 แบบ longest

ยกตัวอย่าง

SSES -> SS

IES -> I

SS -> SS

S ->

ให้คำ CARESSES แปลงเป็น CARESS เพราะSSES เป็นชุดของอักษรที่ยาวที่สุดที่ตรงกับ S1 ในทำนองเดียวกัน คำว่า CARESS ถูกแปลงเป็น CARESS เพราะ S1=`SS' และ CARES แปลงเป็น CARE เพราะ S1=`S' เป็นต้น

อัลกอริทึมที่ได้ทำงานเป็นขั้นตอนดังต่อไปนี้

ขั้น 1 เป็นการจัดการเกี่ยวกับรูปพหูพจน์และคำคุณศัพท์ในรูป -ED แบ่งออกเป็นขั้นต่าง ๆ ดังต่อไปนี้

ขั้น 1a

กฎ	**ตัวอย่าง**
SSES -> SS	caresses -> caress
IES -> I	ponies -> poni ties -> ti
SS -> SS	caress -> caress
S ->	cats -> cat

ขั้น 1b

กฎ	ตัวอย่าง
(m>0) EED -> EE	feed -> feed agreed -> agree
(*v*) ED ->	plastered -> plaster bled -> bled
(*v*) ING ->	motoring -> motor sing -> sing
AT -> ATE	conflat(ed) -> conflate
BL -> BLE	troubl(ed) -> trouble
IZ -> IZE	siz(ed) -> size
(*d and not (*L or *S or *Z)) -> single letter	hopp(ing) -> hop tann(ed) -> tan fall(ing) -> fall hiss(ing) -> hiss fizz(ed) -> fizz
(m=1 and *o) -> E	fail(ing) -> fail fil(ing) -> file

ขั้น 1c

กฎ	ตัวอย่าง
(*v*) Y -> I	happy -> happi sky -> sky

ขั้น 2 เป็นการแปลงรูป S1 ของคำศัพท์ต่าง ๆ ที่ตรงตามเงื่อนไขให้เป็น S2

กฎ	ตัวอย่าง
(m>0) ATIONAL -> ATE	relational -> relate
(m>0) TIONAL -> TION	conditional -> condition rational -> rational
(m>0) ENCI -> ENCE	valenci -> valence
(m>0) ANCI -> ANCE	hesitanci -> hesitance
(m>0) IZER -> IZE	digitizer -> digitize
(m>0) ABLI -> ABLE	conformabli -> conformable
(m>0) ALLI -> AL	radicalli -> radical
(m>0) ENTLI -> ENT	differentli -> different
(m>0) ELI -> E	vileli - > vile
(m>0) OUSLI -> OUS	analogousli -> analogous
(m>0) IZATION -> IZE	vietnamization -> vietnamize
(m>0) ATION -> ATE	predication -> predicate
(m>0) ATOR -> ATE	operator -> operate
(m>0) ALISM -> AL	feudalism -> feudal
(m>0) IVENESS -> IVE	decisiveness -> decisive
(m>0) FULNESS -> FUL	hopefulness -> hopeful
(m>0) OUSNESS -> OUS	callousness -> callous
(m>0) ALITI -> AL	formaliti -> formal
(m>0) IVITI -> IVE	sensitiviti -> sensitive
(m>0) BILITI -> BLE	sensibiliti -> sensible

ขั้น 3 เหมือนกับขั้นตอนที่ 2 คือการแปลงรูป S1 ของคำศัพท์ต่าง ๆ เป็น S2

กฎ	ตัวอย่าง
(m>0) ICATE -> IC	triplicate -> triplic
(m>0) ATIVE ->	formative -> form
(m>0) ALIZE -> AL	formalize -> formal
(m>0) ICITI -> IC	electriciti -> electric
(m>0) ICAL -> IC	electrical -> electric

(m>0) FUL ->	hopeful -> hope
(m>0) NESS ->	goodness -> good

ขั้น 4 แปลงรูปแบบคำเหมือนขั้น 2 และ 3

กฎ	**ตัวอย่าง**
(m>1) AL ->	revival -> reviv
(m>1) ANCE ->	allowance -> allow
(m>1) ENCE ->	inference -> infer
(m>1) ER ->	airliner -> airlin
(m>1) IC ->	gyroscopic -> gyroscop
(m>1) ABLE ->	adjustable -> adjust
(m>1) IBLE ->	defensible -> defens
(m>1) ANT ->	irritant -> irrit
(m>1) EMENT ->	replacement -> replac
(m>1) MENT ->	adjustment -> adjust
(m>1) ENT ->	dependent -> depend
(m>1 and (*S or *T)) ION ->	adoption -> adopt
(m>1) OU ->	homologou -> homolog
(m>1) ISM ->	communism -> commun
(m>1) ATE ->	activate -> activ
(m>1) ITI ->	angulariti -> angular
(m>1) OUS ->	homologous -> homolog
(m>1) IVE ->	effective -> effect
(m>1) IZE ->	bowdlerize -> bowdler

ขั้น 5 เป็นขั้นตอนสุดท้ายในการแปลงรูปคำให้เป็นรากคำนั้น ๆ

ขั้น 5a

กฎ	**ตัวอย่าง**
(m>1) E ->	probate -> probat rate -> rate

(m=1 and not *o) E ->	cease -> ceas

ขั้น 5b

กฎ	**ตัวอย่าง**
(m > 1 and *d and *L) -> single letter	controll -> control roll -> roll

อัลกอริทึมนี้ต้องระวังไม่แปลงคำที่มีขนาดสั้นเกินไป ซึ่งโปรแกรมสามสารถตรวจสอบ ได้จากค่า m ยกตัวอย่างเช่น

ชุด A	ชุด B
-------	--------
RELATE	DERIVATE
PROBATE	ACTIVATE
CONFLATE	DEMONSTRATE
PIRATE	NECESSITATE
PRELATE	RENOVATE

โปรแกรมจะแปลงคำในชุด B โดยการตัด หรดดรป -ATE ออก ในขณะที่ชุด A ยังคงเหมือนเดิม ซึ่งชุด B จะถูกแปลงกลายเป็น DERIVATE/DERIVE, ACTIVATE/ACTIVE, DEMONSTRATE/DEMONSTRABLE, NECESSITATE/NECESSITOUS

ยกตัวอย่าง คำว่า

a. GENERALIZATIONS ถูกแปลงรูปในขั้นต่าง ๆ ดังนี้

   GENERALIZATION (Step 1)

   GENERALIZE (Step 2)

   GENERAL (Step 3)

   GENER (Step 4)

b. OSCILLATORS

   OSCILLATOR (Step 1)

   OSCILLATE (Step 2)

   OSCILL (Step 4)

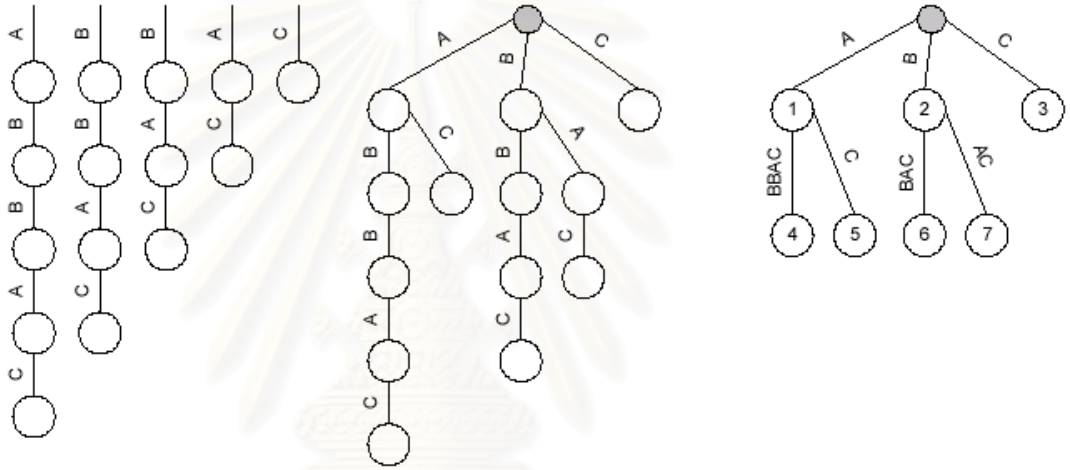
   OSCIL (Step 5)

ภาคผนวก ข

ซัฟฟิกทรี (Suffix Tree)

โครงสร้างต้นไม้แบบซัฟฟิกทรี (Suffix Tree)

ซัฟฟิกทรีเป็นโครงสร้างข้อมูลที่สนับสนุนการจับคู่ตัวอักษรและการค้นหาข้อมูลซึ่งถูกใช้กันอย่างกว้างขวางในระบบการค้นหาข้อมูลต่าง ๆ ยกตัวอย่างการสร้างซัฟฟิกทรีแบบง่ายของ ชุดอักขระ “abbac” โดยแสดงการสร้างเส้นทาง (path) ต่าง ๆ การรวมเส้นทางของซัฟฟิกทรี เข้าด้วยกันและการทำการยุบเส้นทางหนึ่ง ๆ จากรากไปจนถึงโหนดหนึ่งในต้นไม้ (path compression) ดังรูปที่ ข.1



- เส้นทางของ suffix ทั้งหมดใน
- ซัฟฟิกเทียร์ (รวมเส้นทาง suffix ต่าง ๆ เข้าด้วยกัน)
- ซัฟฟิกทรี (ยุบเส้นทางที่มีลูกเดียวรวมกัน)

รูปที่ ข.1 ตัวอย่างการสร้างซัฟฟิกทรี

ในการอธิบายความหมายของซัฟฟิกทรีต้องเกี่ยวข้อง กับ คำสำคัญต่าง ๆ ดังต่อไปนี้

- **โหนดใบ** (Leaf Node) คือโหนดในต้นไม้ที่ไม่มีลูกหลาน จากรูปที่ 3.1 ใบประกอบด้วยโหนด 3 ,4,5,6 และ 7
- **โหนดชัดเจน** (Explicit Node) คือ โหนดในต้นไม้ที่ไม่ใช่โหนดใบ จากรูปที่ 3.1 มีโหนด 1 และ 2 เป็นโหนดชัดเจน
- **โหนดโดยนัย** (Implicit Node) คือ โหนดในต้นไม้ที่อยู่ในซัฟฟิกทรี แต่ถูกกำจัดออก เนื่องจากการรวมเส้นทาง (path) ตั้งแต่จุดหนึ่งที่มีเส้นทางเดียวจนถึงใบ เพราะแต่ละ

โหนดภายในทางนั้นมีโหนดลูกโหนดเดียว จากตัวอย่างในรูปที่ 1 มี prefix เช่น “abb” ที่มีเส้นทาง มาสิ้นสุดที่โหนดโดยนัย

- **ซัฟฟิก (Suffix)** ของชุดตัวอักขระ $\{a_1a_2a_3.....a_n\}$ หมายถึง ลำดับตัวอักษรใด ๆ $\{a_ka_{k+1}a_{k+2}.....a_n\}$ ที่ซึ่ง $1 \leq k \leq n$ และ ตัวอักษร a_i ใด ๆ สามารถนำไปเปรียบเทียบความเหมือนกับตัวอักษร อื่น ๆ และถูกจัดลำดับความสำคัญได้ ยกตัวอย่างซัฟฟิกของชุดอักขระ “abbac” ก็คือ “abbac” “bbac” “bac” “ac” และ “c” เป็นต้น
- **เทียร์ (Trie)** เป็นโครงสร้างต้นไม้ชนิดหนึ่งที่แต่ละโหนดมีจำนวนก้านได้เท่ากับจำนวนของตัวอักษรในชุดอักขระ
- **ซัฟฟิกเทียร์ (Suffix Trie)** เป็นเทียร์ (Trie) ที่จากสร้างจากชุดอักขระ S และสามารถ หาซัฟฟิก (Suffix) ใด ๆ ในชุด อักขระนี้ได้เริ่มจากรากถึงใบของต้นไม้ ดังแสดงในรูป 1 ข

  นิยาม Suffix Trie T สำหรับชุดอักขระ S ที่ประกอบด้วยตัวอักษร m ตัว หมายถึง ต้นไม้ที่มีโหนดใบ m โหนดตั้งแต่ 1 ถึง m แต่ละโหนดภายใน (internal node) ที่ไม่ใช่รากมีลูกอย่างน้อย 2 และแต่ละด้านจากโหนดใด ๆ ถึงลูกของมันจะมีฉลากที่ระบุ โดยตัวอักษรภายในชุดอักขระ S โดยไม่มีด้านใดที่ออกจากโหนดเดียวกันมีฉลาก เหมือนกัน และสำหรับใบ i ใด ๆ จะหมายถึงซัฟฟิก (Suffix) ของ S เริ่มจากตำแหน่งที่ i ที่ปรากฏเป็นฉลากจากรากจนถึงใบนั้น
- **ซัฟฟิกทรี (Suffix Tree)** คือ Suffix Tier ที่ลดโหนดที่มีลูกเดียวออกไป (path compression) และรวมฉลากที่มาถึงโหนดนั้นเข้ากับฉลากที่ออกจากโหนดนั้นเป็นฉลากเดียวกันดังแสดง ในรูป 1 ค

อัลกอริทึมการสร้างซัฟฟิกทรี

อัลกอริทึมดั้งเดิมของการสร้าง Suffix Tree ของชุดอักขระ S ซึ่งมีตัวอักษร m ตัวใช้เวลา $O(m^2)$ แสดงได้ดังนี้

1. Let N_i denote the intermediate tree that encodes all the suffixes from 1 to i
2. Tree N_1 consists of a single edge between the root of the tree and a leaf labeles
   1. The edge is labeled with the string S.
3. Tree N_{i+1} is constructed from N_i as fllows :
   1.1 Start at the root of N_i the อัลกอริทึม finds the longest path from the root whose label matches a prefix of S[i+1..m]. This path is found by successively comparing and matching words in suffix S[i+1..m] to words along a unique path from the root,until no further matches are possible.
   1.2 When no further matches are possible , the อัลกอริทึม is either at a node , say w , or it is in a middle of an edge.
   1.3 If it is in the middle of an edge , say (u,v) , then it breaks (u,v) into two wdges by inserting a new node w just after the last word on the edge whose label matches a prefix of the suffix S[i+1..m].
   1.4 In either case , the อัลกอริทึม creates a new egde (w,i+1) running from w to a new leaf labeled i+1 , and labels the new edge with the unmatched part of suffix S[i+1..m].

รูปที่ ข.2 อัลกอริทึมการสร้างซัฟฟิกทรี (Suffix Tree)

อัลกอริทึมดังรูปที่ 2 ยังมีข้อเสียคือต้องสร้างซัฟฟิกทรี (Suffix Tree) แบบ reverse order กล่าวคือต้องทราบชุดอักขระในการสร้างซัฟฟิกทรี (Suffix Tree) นั้นทั้งหมดเสียก่อน เพราะ อัลกอริทึมนี้เริ่มสร้างซัฟฟิกทรี (Suffix Tree) จากชุดอักขระทั้งหมดแล้วค่อย ๆสร้างต้นไม้ให้ สมบูรณ์จากซัฟฟิก (suffix) ของชุดอักขระ S จนครบ อัลกอริทึมนี้จึงไม่เหมาะกับระบบการทำงาน แบบออนไลน์ เช่น ระบบค้นคืนสารสนเทศ

ต่อมา Esko Ukkonen [11] ได้ทำการพัฒนาอัลกอริทึมซึ่งทำงานกับชุดอักขระจาก ซ้ายไปขวา กล่าวคือการสร้างซัฟฟิกทรีจาก prefixes ของชุดอักขระ S ยกตัวอย่างชุดอักขระ “abbac”

จะถูกนำ ไปสร้าง Suffix Tree โดยการใส่ prefixes ต่าง ๆ ลงไปตามลำดับคือ “a” , “ab” , “abb” , “abba” และ “abbac”

อัลกอริทึมของ Ukkonen เริ่มจากรากแล้วจึงเพิ่ม prefix ของชุดอักขระเข้าไปในต้นไม้ ทีละตัว จนได้ต้นไม้ที่สมบูรณ์เมื่อ prefix ตัวสุดท้ายถูกใส่เข้าไปซึ่งก็ชื่อชุดของอักขระทั้งหมด นั่นเอง การทำงานของอัลกอริทึมนี้แสดงได้ดังนี้

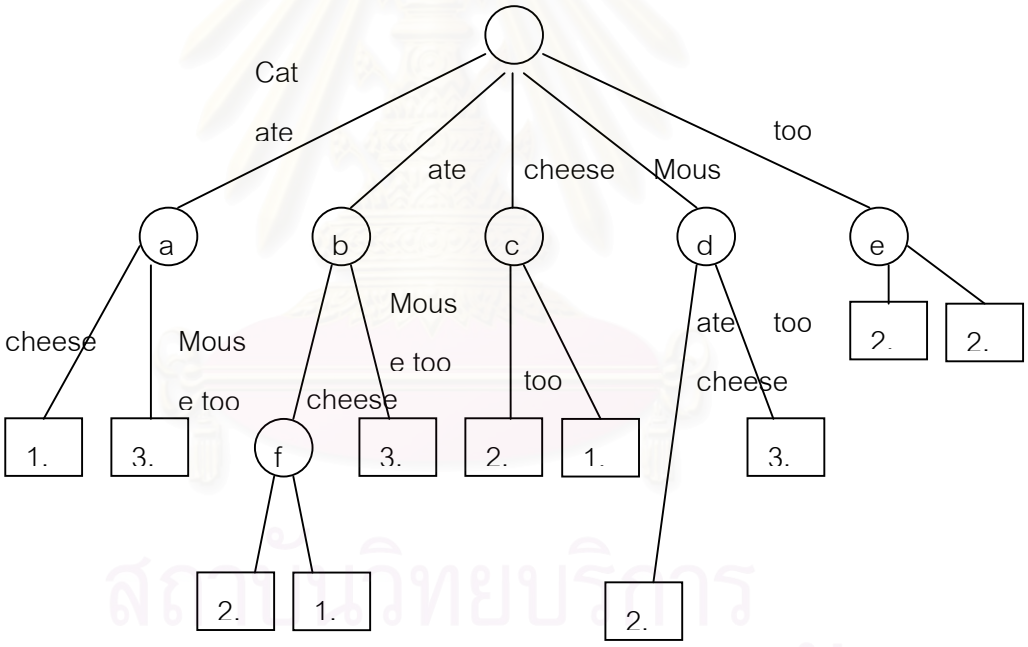
```
Update (new_prefix)
{       current_suffix = active_point
        test_char = last_char in new_prefix
        done = false
        while (!done)
        {       If current_suffix ends at an explicit node
                {       If the node has no descendant edge starting with test_char
                                Create new leaf edge starting at the explicit node
                        Else
                                Done = true
                } else
                {       if the implicit's node next char is not test_char
                        {       split the edge at the implicit node
                                create new leaf edge starting at the split in the edge
                        } else
                        done = true
                }
                if current_suffix is the empty string
                        done = true
                else
                        current_suffix = next_smaller_suffix(current_suffix)
        } /*end while loop*/
        active_point = current_suffix
} /*end อัลกอริทึม*/
```

รูปที่ ข.3 อัลกอริทึมการสร้างซัฟฟิกทรี (Suffix Tree) ของ Ukkonen

เจเนอรัลไลซ์ซัฟฟิกทรี (Generalized Suffix Tree - GST)

เจเนอรัลไลซ์ซัฟฟิกทรี (GST) คือ กลุ่มของชุดอักขระหลาย ๆ ชุด ที่ถูกนำไปสร้างเป็นซัฟฟิกทรี (Suffix Tree) ซึ่ง suffix ของชุดอักขระใด ๆ จะปรากฏเป็นฉลากอยู่ภายในต้นไม้ เริ่มจากรากจนถึงใบ

นิยาม เจเนอรัลไลซ์ซัฟฟิกทรี T สำหรับกลุ่มของชุดอักขระ S จำนวน n ชุด $\{S_1,S_2..S_n)$ แต่ละชุดอักขระมีตัวอักษร m_n ตัว เป็นต้นไม้ที่มีใบเท่ากับจำนวนอักษรทั้งหมดของชุดอักขระทุกชุด รวมกันและสามารถเขียนสัญลักษณ์แสดงความหมายของแต่ละใบเป็น (k,l) เมื่อ k หมายถึง ลำดับที่ของชุดอักขระ(1ถึง n) และ l หมายถึงตำแหน่งของตัวอักษรภายใน ชุดอักขระที่ k (1 ถึง m_k) แต่ละโหนดภายใน (internal node) ที่ไม่ใช่รากมีลูกอย่างน้อย 2 และแต่ละด้านจากโหนดใด ๆ ถึงลูกของมันจะมีฉลากที่ระบุโดยตัวอักษร ภายในชุดอักขระ S โดยไม่มีด้านใดที่ออกจาก โหนดเดียวกันมีฉลากเหมือนกัน สำหรับใบ (i,j) ใด ๆ หมายถึง Suffix ที่เกิดจากการรวมกัน ของฉลาก จากรากจนถึงใบของชุดอักขระที่ S_i เริ่มจากตัวอักษร ตัวที่ j นั่นคือ $S_i[j...m_i]$



รูปที่ ข.4 ตัวอย่างของ เจเนอรัลไลซ์ซัฟฟิกทรี ของชุดอักขระ 3 ชุดคือ “cat ate cheese” , mouse ate cheese too” และ “cat ate mouse too”

ภาคผนวก ค

ตัวอย่างข้อมูลในการทดลองและรหัสคำ

คำภาษาไทย	รหัสคำที่ได้จากการเข้ารหัสคำภาษาไทย	คำภาษาอังกฤษ	รหัสคำที่ได้จากการเข้ารหัสคำภาษาอังกฤษ
ซิลเวอร์	slviW	Silver	slviW
ปิโตรเลียม	ptrlmiol	petroleum	ptrlmiol
เอนจิเนียร์	njneil	engineer	njneil
อิเล็กทรอนิกส์	lktrnkie$i	electronic	lktrnkie$i
คอมพิวเตอร์	kmpt$XW	computer	kmpt$XW
เทคโนโลยี	tknlyeooE	technology	tknljeooE
พริ้นเตอร์	prntiW	printer	PrntiW
มอนิเตอร์	mnt$iW	monitor	mnt$iW
อินเตอร์เฟส	ntfsiWe	interface	ntfsiW
แวบ	vbw	wap	vpw
ช็อกโกแลต	ckklt$ow	chocolate	ckltooe
ซูเปอร์มาร์เก็ต	spmktUWae	supermarket	spmktUWae
โนเกีย	nkol	nokia	nkol
สนุกเกอร์	snkkuW	snooker	snkUw
อิริคสัน	rksniiV	Ericson	rksneiV
เปอร์เซ็นต์	psnWe	percent	psnWe
ทรานซิสเตอร์	trnsstaiW	transistor	trnsstaiW
ซุป	spu	soup	spu
โบนัส	bnsoV	bonus	bnsoV
ทาวน์เฮาส์	txhR	townhouse	thxR
ไมโครโฟน	mkrfn!oo	microphone	mkrfn!oo
แก๊ส	ksw	gas	ksw
เนกไท	nkte!	necktie	nkte!
นิวเคลียร์	nklXl	nuclear	nklXl

คำภาษาไทย	รหัสคำที่ได้จากการเข้ารหัสคำภาษาไทย	คำภาษาอังกฤษ	รหัสคำที่ได้จากการเข้ารหัสคำภาษาอังกฤษ
อินฟราเรด	nfrrdiae	infrared	nfrrdiae
กิโลกรัม	klkrmioV	kilogram	klkrmiow
แบดมินตัน	bdmntnwiV	badminton	bdmntnwiV
จุฬาลงกรณ์	jvlgkrua	chulalongkorn	kllgknU$$
ไทย	t!	thai	t!
คลินตัน	klntniV	clinton	klntniV
โตโยต้า	tytooa	toyota	tytooa
แมคโดนัลด์	mkdnloV	mcdonald	mkdnloV
เซนทรัล	sntrleV	central	sntrleV
โค้ก	kko	coke	kko
มิตซูบิชิ	mtsbciUii	mitsubishi	mtsbciUii
ยามาฮ่า	ymhaaa	yamaha	Ymhaaa
ฮานามิ	hnmaai	hanami	hnmaai
เอสโซ่	sseo	esso	sseo
พานาโซนิค	pnsnkaaoi	panasonic	pnsnkaaoi
ซิงเกอร์	sgkiW	singer	SgkiW

จากการทดลองพบว่าคำทับศัพท์บางคำเมื่อเข้ารหัสคำภาษาไทยแล้วจะได้รหัสคำต่างจากรหัสคำที่ได้จากการเข้ารหัสคำภาษาอังกฤษ โดยเฉพาะกรณีที่คำนั้นเป็นชื่อเฉพาะของคน

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวกฤษณี อริยชาญศิลป์ เกิดวันที่ 13 พฤศจิกายน พ.ศ. 2517 ที่กรุงเทพมหานคร สำเร็จการศึกษาวิทยาศาสตร์คอมพิวเตอร์ จากจุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2539 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตร์มหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมศาสตร์ ที่จุฬาลงกรณ์ มหาวิทยาลัย ในปีการศึกษา 2542