นนทรีเสิร์จ : เสิร์จเอ็นจินสำหรับแคมปัสเน็ตเวอร์ค
NontriSearch : Search Engine For Campus Network

สุรศักดิ์ สงวนพงษ์ สุทธิพล วรางฤทธิ์
E-mail : { nguan, g4165234 }@ku.ac.th

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

บทคัดย่อ

นนทรีเสิร์จเป็นระบบเสิร์จเอ็นจินเพื่อใช้ค้นหาคำในเครือข่ายเวบ การทำงานของนนทรีเสิร์จใช้หลักการจัดทำดรรชนีและมีส่วนสอบถามคำที่ต้องการเพื่อสืบค้นคำในในฐานข้อมูลในรูปแบบเดียวกับเวบโรบอต ความสามารถในการสืบค้นของนนทรีเสิร์จทำได้ทั้งคำเต็มในภาษาไทยและภาษาอังกฤษ และสนับสนุนการสืบค้นโดยกำหนดเงื่อนไขบูลีน กลไกอ่านโฮมเพจเพื่อจัดทำดรรชนีทำงานได้ตามข้อกำหนดของมาตรฐานข้อยกเว้นของโรบอต (Standard Robot Exclusion) ระบบได้รับการออกแบบให้มีขนาดเล็กเพื่อความเหมาะสมต่อการใช้งานเป็นเสิร์จเอ็นจินสำหรับแคมปัสเน็ตเวอร์คหรืออินทราเน็ตในองค์กร

1. บทนำ

อินเทอร์เน็ตในปัจจุบันมีข้อมูลข่าวสารอยู่มากมาย โดยเฉพาะข้อมูลไฮเปอร์เท็กซ์ที่มีให้บริการในระบบ เวบ เป็นที่นิยมใช้อย่างแพร่หลาย การค้นหาข้อมูลที่มีเนื้อหาที่ตรงกับความสนใจจากเวบที่มีหลายล้านเพจในอินเทอร์เน็ตจึงเป็นเรื่องยาก หากไม่มีเครื่องมือช่วยค้นหาซึ่งที่รู้จักกันดีในชื่อของ เสิร์จเอ็นจิน (Search Engine)

บริการค้นหาเวบเพจที่มีอยู่ในปัจจุบันเช่น AltaVista, Lycos, Yahoo, HotBot, InfoSeek [1] ฯลฯ ซึ่งมักเรียกโดยรวมทั้งหมดว่าเป็นเสิร์จเอ็นจินนั้น แท้ที่จริงแล้วมีบางแห่งมีกลไกทำงานแบบเสิร์จเอ็นจินอย่างแท้จริง และบางแห่งทำงานด้วยวิธีเก็บเวบเพจไว้เป็น *ไดเรคทอรี* ซึ่งมีความแตกต่างกันโดยหลักพื้นฐานการสร้างดรรชนีเวบเพจ

ไดเรคเทอรีมีการทำงานที่แตกต่างกับเสิร์จเอ็นจิน คือใช้การปรับเพิ่มข้อมูลโดยผู้ดูแลระบบเองโดยไม่ได้ทำอย่างอัตโนมัติ เวบไซต์ใดที่ต้องการมีรายชื่อในไดเรคเทอรีต้องติดต่อไปยังผู้ดูแลไดเรคเทอรี เพื่อให้ผู้ดูแลจำแนกและจัดเก็บลงฐานข้อมูล ระบบการเก็บแบบไดเรคเทอรีอาจให้ผลลัพธ์การค้นหาข้อมูลที่ตรงประเด็นมากกว่าเสิร์จเอ็นจินเพราะผ่านการแยกหมวดหมู่มาแล้ว ตัวอย่างของบริการค้นหาที่ใช้ระบบไดเรคเทอรีได้แก่ Yahoo

สำหรับเสิร์จเอ็นจินโดยความหมายที่แท้จริงแล้วเป็นระบบซอฟต์แวร์ที่มีโปรแกรม *โรบอต* [2] ทำหน้าที่อ่านเวบเพจจากไซต์ต่างๆโดยอัตโนมัติ จากนั้นจึงนำเวบเพจที่อ่านได้มาทำดรรชนี เสิร์จเอ็นจินจะตรวจสอบลิงค์ในแต่ละหน้าของเวบเพจเพื่อเข้าไปทำดรรชนีของเวบเพจนั้นต่อไปอีก ตัวอย่างของเสิร์จเอ็นจินนี้ได้แก่ HotBot หรือ Alta Vista หรือ Infoseek เป็นต้น

การค้นหาข้อมูลภาษาไทยในเวบเพจยังคงเป็นความต้องการหลักประการหนึ่ง โดยเฉพาะอย่างยิ่งในองค์กรที่ใช้ระบบอินทราเน็ตภายในโดยมีบริการเวบ และต้องการเสิร์จเอ็นจินที่ให้บริการเฉพาะการค้นหาข้อมูลภายในที่ตอบสนองได้ทั้งการสืบค้นข้อมูลภาษาไทยและอังกฤษ ในปัจจุบันเสิร์จเอ็นจินส่วนใหญ่ไม่สนับสนุนการทำดรรชนีภาษาไทยเนื่องจากต้องอาศัยการทำงานเพิ่มเติมด้านการวิเคราะห์คำ บริการค้นหาคำภาษาไทยที่พบได้ในปัจจุบันอาจมีเพียง AltaVista [3] ที่ดัดแปลงให้สามารถทำดรรชีภาษาไทยได้ แต่ก็จัดอยู่ในกลุ่มของเสิร์จเอ็นจินขนาดใหญ่สำหรับเครือข่ายเวบที่ครอบคลุมทั้งโลก

นนทรีเสิร์จเป็นเสิร์จเอ็นจินแบบโรบอต มีกลไกการทำงานแบบสร้างดรรชนีคำทั้งในภาษาไทยและภาษาอังกฤษตามหลักการแบบ *ดรรชนีผกผัน* (inverted index) [4] และจับเก็บคำโดยใช้วิธีแฮชชิง รูปแบบของดรรชนีผกผันช่วยให้สามารถค้นหาคำได้อย่างรวดเร็วเพราะไม่มีกระบวนการค้นหาคำจากข้อมูลต้นฉบับใดๆ หากมีแต่เพียงการสืบค้นจากตารางเก็บดรรชนีเท่านั้น ตัวอย่างเช่นนนทรีเสิร์จสามารถใช้เวลาซีพียูโดยเฉลี่ยเพียง 2 วินาทีสำหรับสืบค้นคำในการค้นหาคำที่จากเวบเพจประมาณ 11,000 หน้าที่มีความจุรวมราว 100 เมกะไบต์

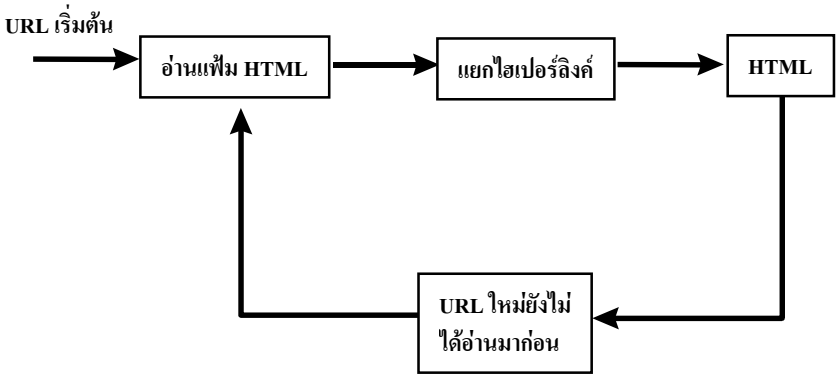
2. โครงสร้างของนนทรีเสิร์ช

โครงสร้างของนนทรีเสิร์จแบ่งออกเป็น 3 ส่วนหลัก คือ

1) นนทรีบอต
2) อินเด็กเซอร์
3) นนทรีเอ็นจิน

2.1 นนทรีบอต

ทำหน้าที่เป็นโรบอตอ่านเวบไซต์ๆและจัดเก็บเพื่อส่งต่อให้ อินเด็กเซอร์สร้างฐานข้อมูลต่อไป รูปที่ 1 แสดงโครงสร้างการทำงานของนนทรีบอตซึ่งจะอ่านเวบเพจและจัดเก็บโดยจำลองโครงสร้างของไดเรคทอรีเช่นเดียวกับตำแหน่งที่เก็บของแฟ้มต้นฉบับในเวบไซต์นั้น โรบอตจะสร้างคิวของเวบเพจที่ต้องการทำดรรชนีโดยกำหนดเพจเริ่มต้นให้ในคิว หากมองโครงสร้างของเวบเพจเสมือนโครงสร้างต้นไม้การท่องเวบเพจจะเป็นแบบแนวกว้าง (breath first) โดยสามารถเลือกได้ว่าจะจำกัดระดับความลึกในการอ่านเวบเพจหรือไม่



รูปที่ 1 การทำงานของนนทรีบอต

2.2 อินเด็กเซอร์

ทำหน้าที่เป็นตัวสร้างดรรชนี ผลลัพธ์จากอินเด็กเซอร์ประกอบด้วยฐานข้อมูล 3 ฐานข้อมูล

- index.idx เป็นฐานข้อมูลที่บรรจุคำศัพท์ที่พบทั้งหมด และหมายเลขกำกับ URL ที่เป็นตัวชี้ไปยังเพจที่มีคำศัพท์อยู่ โครงสร้างของฐานข้อมูลนี้แสดงได้ดังตารางที่ 1
- url.idx เป็นฐานข้อมูลที่เก็บหมายเลข URL และ ชื่อ URL โครงสร้างของฐานข้อมูลนี้แสดงได้ดังตารางที่ 2
- title.idx เป็นฐานข้อมูลที่เก็บหมายเลขกำกับ URL และคำบรรยายของเวบเพจนั้น โดยเก็บหัวเรื่องของเวบเพจจากแท็ก <TITLE> และอักขระ 250 ตัวแรกของเวบเพจ ตารางที่ 3 แสดงโครงสร้างของฐานข้อมูลนี้

ตารางที่ 1 โครงสร้างภายในฐานข้อมูล index.idx

Key	Content
glance	1|2|3|4
gland	3|4|5|7|11
glare	5|7|8|9|13|120
glass	1|2|3|4|5
glaze	12|15|18|20|23|12
gleam	1|2|3|4|5|6|7|8|9
glean	2|4|5|6|8|10|12|14

glee	1|3|5|7|9|11|13|15
glen	1|2|3|4|5|6|7|8|9|1

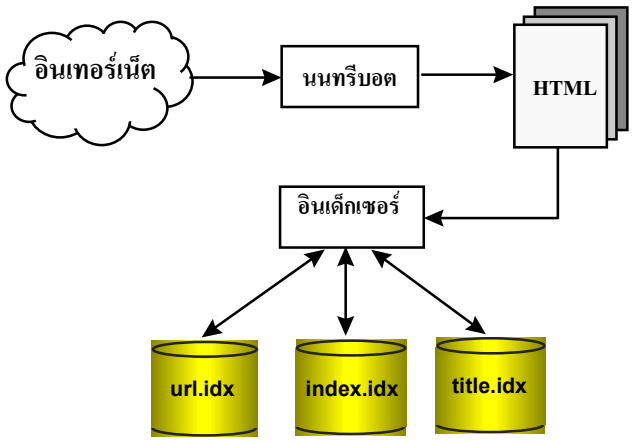
ตารางที่ 2 ข้อมูลภายในฐานข้อมูล index.idx

Key	Content
1	www.ku.ac.th
2	www.cpe.ku.ac.th/search.html
3	std.cpc.ku.ac.th/~b37spw/index.html
4	www.sci.ku.ac.th/
5	std.cpc.ku.ac.th/user.html
6	www.eng.ku.ac.th/~webman
7	www.ee.ku.ac.th/
8	is1.eng.ku.ac.th
9	www.eng.ku.ac.th/~webman/resources
10	www.lib.ku.ac.th/books

ตารางที่ 3 ข้อมูลภายในฐานข้อมูล title.idx

Key	Content
1	มหาวิทยาลัยเกษตรศาสตร์xxxxxxxxxxx.xxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
2	คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ xxxxxx xxxxxxx xxxxxxx xxxxxxxxxx xxxx xxxx
3	Welcome to Department of Computer Engineering xxxx xx xxxxx xxxxxx xxxxxx xxxxxxxxxxxxxxxxxxxxxxxxx
4	Nectec homepage xxxxxxxxx xxxxxxxxx xxxxxxxxx xxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

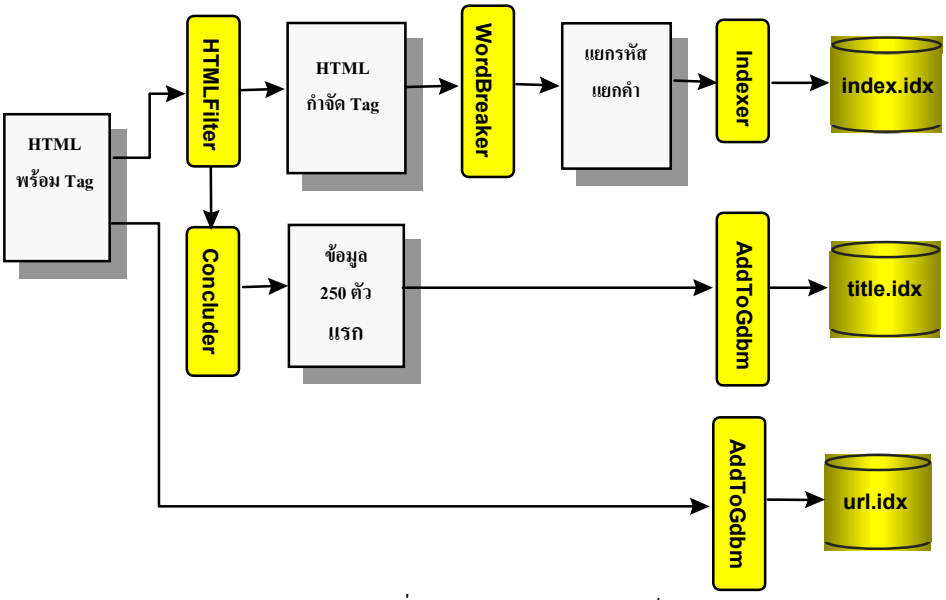
ฐานข้อมูลทั้ง 3 ฐานข้อมูลเป็นฐานข้อมูลชนิด GDBM (GNU Database Management) โครงสร้างความสัมพันธ์ของฐานข้อมูลและอินเด็กเซอร์แสดงได้ดังรูปที่ 2



รูปที่ 2 ฐานข้อมูลจากอินเด็กเซอร์

โครงสร้างการทำงานของอินเด็กเซอร์ มีการองค์ประกอบย่อยดังรูปที่ 3 แต่ละส่วนทำหน้าที่ดังนี้

1) HtmlFilter ทำหน้าที่กรอง HTML Tag ออก และหากพบรหัสเอสเคปที่ใช้แทนอักขระ จะดำเนินการแปลงไปเป็นรหัสแอสกี
2) Wordbreaker ทำหน้าที่แบ่งคำๆ โดยตัดคำที่ยาวที่สุดที่เป็นไปได้โดยอาศัยดิคชันนารี
3) Indexer ทำหน้าที่นำคำศัพท์แต่ละคำไปเก็บในสร้างฐานข้อมูล index.idx
4) Concluder ทำหน้าที่หาคำบรรยายเวบเพจโดยนำข้อความจาก <Title> xxxx </Title> มาเป็นหัวเรื่อง และนำอีก 250 ตัวอักษรถัดไปมาเป็นคำบรรยาย
5) AddToGDBM ทำหน้าที่เป็นตัวเพิ่มข้อมูลเข้าไปในฐานข้อมูล url.idx และ title.idx



รูปที่ 3 การทำงานของอินเดกเซอร์

2.3 นนทรีเสิร์จ

เป็นส่วนที่รับคำศัพท์ที่ต้องการค้นหาจากผู้ใช้ผ่านทางซีจีไอมาค้นหาในฐานข้อมูลและส่งผลการค้นหากลับไปแสดงบนหน้าจอ โดยนำคำศัพท์แต่ละคำที่ได้มาเปิดหาในฐานข้อมูล index.idx และนำผลลัพธ์ที่ได้จากการค้นหาของแต่ครั้งมาหาเอกสารที่มีคำศัพท์ครบทุกคำ แล้วค่อยเปิดฐานข้อมูล url.idx และ title.idx เพื่อแสดงชื่อ URL และคำบรรยายออกหน้าจอ

3. ขีดความสามารถและสมรรถนะ

เทคนิคดรรชนีผกผันที่ใช้ในนนทรีเสิร์ชถึงแม้ว่าจะมีจุดอ่อนด้านการใช้เนื้อที่ขนาดใหญ่เพื่อสร้างดรรชนี กล่าวคืออาจมีขนาดของดรรชนีเป็น 50%-300% [6] ของขนาดข้อมูลต้นฉบับ และในกรณีของนนทรีเสิร์จจะมีขนาดแฟ้มดรรชนีราว 185% ของขนาดข้อมูลต้นฉบับ และไม่ต้องเก็บข้อมูลต้นฉบับไว้อีกต่อไป หากแต่ในแง่ความรวดเร็วของการค้นหาแล้ววิธีนี้จะให้ผลลัพธ์ที่รวดเร็วตรงกับรูปแบบการให้บริการในระบบเครือข่ายที่ต้องตอบสนองการให้บริการแก่ผู้ใช้จำนวนมาก

การสร้างดรรชนีสามารถเลือกให้ไม่ต้องสร้างดรรชนีของคำในกลุ่ม “Stop word” เช่น a, an the, ก็, นี้ เป็นต้น เพื่อช่วยลดขนาดของแฟ้มดรรชนี คำที่เหล่านี้สามารถเพิ่มหรือลดได้ตามความต้องการ

หากมีการสอบถามหาคำโดยใช้เงื่อนไขบูลีนเช่นเมื่อมีข้อคำถาม ***คำ1*** **AND** ***คำ2*** นนทรีเสิร์จจะค้นหารายการ URL ที่พบของแต่ละคำ จากนั้นจึงหาหมายเลข URL ร่วมกันเพื่อกำหนดเพจที่บรรจุคำทั้งสอง สัญลักษณ์ที่ใช้กำหนดฟังก์ชันบูลีนคือ

& (AND)

+ (OR)

- (NOT)

ตัวอย่างเช่น 'ไทย &กรุงเทพ–อเมริกา' หมายถึง ค้นหาเอกสารที่มีคำว่า 'ไทย' และ 'กรุงเทพ' แต่ไม่มีคำว่า 'อเมริกา' หากไม่มีการระบุเครื่องหมายจะใช้เครื่องหมาย & แทนให้โดยอัตโนมัติ สำหรับการค้นหาภาษาไทย ถ้าค้นหาเป็นประโยค นนทรีเสิร์จะตัดประโยคที่ค้นหาออกเป็นคำๆ ให้โดยอัตโนมัติ

นนทรีเสิร์จสนับสนุนการทำงานตามข้อกำหนดของ Standard Robot Exclusion [5] ซึ่งเป็นมาตรฐานที่ช่วยให้ผู้ดูแลเซอร์ฟเวอร์กำหนดได้ว่าจะแนะนำให้โรบอตดำเนินการอ่านเพจเพื่อนำไปสร้างดรรชนีหรือไม่

สำหรับสมรรถนะการค้นหาคำจากเวบเพจจำนวนประมาณ 11,000 หน้า ขนาด 104 เมกะไบต์ ทำงานบนเครื่องเพนเทียมโปร 200 หน่วยความจำขนาด 32 เมกะไบต์ ฮาร์ดดิสก์แบบ IDE ทำงานภายใต้ระบบปฏิบัติการไลนักซ์ การค้นหาคำที่รับคำศัพท์ผ่านทางซีจีไอใช้เวลา 0.6-5 วินาที ตัวอย่างเช่นการค้นหาคำว่า *zurich* มีรวมกัน 10 แห่ง ใช้เวลาค้น 0.8 วินาที การค้นหาคำว่า *ประกาศ* มีรวมกัน 150 แห่ง ใช้เวลาค้น 3 วินาที แต่หากทดลองหาคำศัพท์โดยเปลี่ยนวิธีรับคำ แทนที่จะรับคำศัพท์จากซีจีไอ มาเป็นรับคำศัพท์จากไฟล์ของคำศัพท์ที่จะค้นหา และเปลี่ยนวิธีแสดงผลลัพธ์จากซีจีไอ มาเป็นหน้าจอ จะได้ผลการทดลองดังตารางที่ 4

ตารางที่ 4 ผลการทดลองเวลาที่ใช้ในการค้นหาคำศัพท์ (วินาที)

	User time	System time	Elapsed time
1 คำ	0.01	0.02	0.21
10 คำ	0.08	0.27	0.39
100 คำ	0.20	2.05	2.5
1000 คำ	3.9	19.56	38.36
10000 คำ	28.97	191.3	306.87

user time คือเวลาที่ใช้การทำงานของโปรแกรม ซึ่งไม่นับเวลาที่ใช้ในการเรียก System call

system time คือเวลาที่ใช้ในการทำงานของ System call

elapsed time คือเวลาที่ใช้จริงทั้งหมด ซึ่งรวมเวลาที่ทำงานให้กับโปสเซสอื่นด้วย

4. พัฒนาการในอนาคต

การค้นหาคำในขณะนี้ยังคงจำกัดด้วยการสะกดคำที่ถูกต้อง พัฒนาการในขั้นต่อไปคือการตรวจสอบคำที่มีการสะกดใกล้เคียงและการสะกดแบบพ้องเสียง [7]

การทำดรรชนีใหม่เมื่อเพจเกิดเปลี่ยนแปลงสามารถทำได้โดยไม่ต้องทำดรรชนีซ้ำใหม่ทั้งหมด รูปแบบการทำดรรชนีแบบเพิ่มขณะนี้กำลังอยู่ระหว่างดำเนินการปรับแต่งเพิ่มเข้าสู่นนทรีเสิร์จ นอกจากนี้ยังมีโครงการออกแบบระบบแจ้งการปรับปรุงเพจกลับมายังตัวสร้างดรรชนีเพื่อให้สามารถดำเนินการปรับดรรชนีใหม่นอกเหนือไปจากการปรับปรุงตามกำหนดเวลาปกติ

ค่าเผื่อเลือกย่อยเพื่ออำนวยความสะดวกและให้ผู้ใช้เลือกกำหนดกลุ่มเป้าหมายของเวบเพจเช่นวันเวลาย้อนหลัง การเลือกค้นเฉพาะเพจในโดเมนกำหนดให้ หรือการค้นหาเอกสารเฉพาะในยูสเน็ต เหล่านี้จะอยู่ในแผนการดำเนินการขั้นต่อไป

5. เอกสารอ้างอิง

[1] Sullivan, D., *The Majors Search Engines, http://seachengineswatch.com/facts/major.html.*

[2] Mauldin, M., *Lycos : Design Choices in the Internet Search Services*, IEEE Expert Vol. 15 No. 5 September 1997.

[3] Altavista Search Network, *http://altavista.digital.com/av/content/av_network.html.*

[4] Gonnet, G.H., Baeza-Yates, R., *Handbook of Algorithms and Data Structures In Pascal and C*, Addison-Wesley Publishing Company Inc., 1991.

[5] Koster, M., *A Standard Robot Exclusion, http:/info.webcrawler.com.*

[6] Faloutos, C., *Access Methods for Text,* ACM Computing Surveys, Vol. 17, 1985.

[7] Ongroongruang, S., et.al., English to Thai Word Retrieval using Sound Index, In *the 2nd Symposium on Nataral Language Processing*, Bangkok, Thailand., 1998