การวิเคราะห์เปรียบเทียบเครื่องมือที่ใช้ในการทำดัชนีข้อมูล

COMPARISION OF FULL-TEXT INDEXING

โดย

สุดาวรรณ์ ธัญญสกุลกิจ

SUDAWAN THANYASAKULKIT

อาจารย์ที่ปรึกษา

ผศ.ดร.พรฤดี เนติโสภากุล

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ภาคฤดูร้อน ปีการศึกษา 2553

COMPARISION OF FULL-TEXT INDEXING

SUDAWAN THANYASAKULKIT

A REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS OF THE COURSE

SYSTEM DEVELOPMENT PROJECT

MASTER OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

SUMMER/ 2010

ใบรับรองโครงการพัฒนาระบบงาน (System Development Project)

เรื่อง

การวิเคราะห์เปรียบเทียบเครื่องมือที่ใช้ในการทำดัชนีข้อมูล

COMPARISION OF FULL-TEXT INDEXING

นางสาวสุดาวรรณ์ ธัญญสกุลกิจ

รหัสประจำตัว 49066530

ขอรับรองว่ารายงานฉบับนี้ ข้าพเจ้าไม่ได้คัดลอกมาจากที่ใด

รายงานฉบับนี้ได้รับการตรวจสอบและอนุมัติให้เป็นส่วนหนึ่งของ

การศึกษาวิชาโครงการพัฒนาระบบงาน หลักสูตรวิทยาศาสตรมหาบัณฑิต (เทคโนโลยีสารสนเทศ)

ภาคฤดูร้อน ปีการศึกษา 2553

.....อาจารย์ที่ปรึกษา

(ผศ.ดร.พรฤดี เนติโสภากุล)

.....กรรมการสอบ

(รศ.ดร.วรพจน์ กรีสุระเดช)

.....กรรมการสอบ

(รศ.ดร.อาริต ธรรมโน)

หัวข้อ	การวิเคราะห์เปรียบเทียบเครื่องมือที่ใช้ในการทำดัชนีข้อมูล
นักศึกษา	นางสาวสุดาวรรณ์ ธัญญสกุลกิจ
รหัสนักศึกษา	49066530
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2553
อาจารย์ที่ปรึกษา	ผศ.ดร. พรฤดี เนติโสภากุล

บทคัดย่อ

ในยุคที่เทคโนโลยีเข้ามามีบทบาท ทำให้ข้อมูลข่าวสาร องค์ความรู้ต่าง ๆ มีมากมายมหาศาล การที่จะค้นหาข้อมูลจำนวนมากจำเป็นต้องอาศัยเครื่องมือที่ช่วยในการค้นหาที่เรียกว่า search engine ปัจจุบัน search engine ส่วนใหญ่ใช้วิธีการสร้างดัชนีของข้อมูลเอาไว้นำทาง พาไปยังแหล่งข้อมูลจริง เพื่อความรวดเร็วและแม่นยำ ส่วนสำคัญของการค้นหาคือการสร้างดัชนีข้อมูลขึ้นมาแล้วดูว่าดัชนีนั้น ๆ มีประสิทธิภาพแค่นั้น สามารถตอบสนองถึงความต้องการของผู้ใช้ได้อย่างทันทีหรือไม่

การจะเลือกหยิบใช้เครื่องมือที่ใช้ในการค้นหามาซักหนึ่งอัน จึงมีความสำคัญ ในโครงงานนี้จึงได้หยิบเอาเครื่องมือมาสองชนิดที่ได้รับความนิยมได้แก่ Apache Lucene และ Sphinx มาทำการเปรียบเทียบถึงประสิทธิภาพในการทำดัชนี โดยจะพิจารณาถึง เวลาที่ใช้ในการสร้างดัชนี พื้นที่ที่ใช้เก็บดัชนีที่ถูกสร้าง และคุณภาพของดัชนี โดยใส่คำค้นชนิดต่าง ๆ แล้วดูว่าผลที่ได้จากการค้นหานั้นมากน้อยเพียงใดตรงกับความต้องการของผู้ใช้หรือไม่

Title	Comparison for Full – Text Indexing Tool
Student	Miss. Sudawan Thanyasakulkit
Student ID.	49066530
Degree	Master of Science
Program	Information Technology
Major	Information Science
Academic Year	2010
Advisor	Asst.Prof.Dr.Poorudee Netisopakul

ABSTRACT

Today, the term information technology is more recognizable regards with many aspects of computing and technology. Search Engine is one of the method that has been accepted This Project choose Apache Lucene API and Sphinx API for test .Lucene is a Java library for creating and searching through a full text index . Sphinx is a full-text search engine, distributed under GPL version 2. Commercial licensing (eg. for embedded use) is also available upon request

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงได้ด้วยดี ด้วยคำแนะนำ และให้คำปรึกษา ตลอดจนการตรวจสอบแก้ไข เพื่อให้โครงงานนี้เสร็จสมบูรณ์ จาก ผศ.ดร.พรฤดี เนติโสภากุล ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน

คณาจารย์คณะเทคโนโลยีสารสนเทศทุกๆ ท่าน ที่ได้ให้ความรู้มาโดยตลอด

ขอบคุณเจ้าหน้าที่ประจำคณะเทคโนโลยีสารสนเทศทุกท่าน ที่อำนวยความสะดวกในด้านต่างๆ

ขอบคุณ คุณสุนทร และ คุณสุนิสา พี่ชาย-พี่สาว ที่ช่วยกวดขัน ผลักดัน ให้โครงการสำเร็จลุล่วงไปได้ด้วยดี

สุดท้ายนี้ขอขอบพระคุณ บิดา มารดา และพี่ๆ ที่ให้กำลังใจมาโดยตลอด

สุดาวรรณ์ ธัญญสกุลกิจ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูปที่ **หน้า**

สารบัญตาราง

ตารางที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในโลกยุคปัจจุบันเทคโนโลยีเริ่มเข้ามามีบทบาทมากขึ้น องค์กรต่าง ๆ เล็งเห็นถึงความสำคัญที่จะนำเอาเทคโนโลยีไปใช้ เพิ่มผลผลิต ลดค่าใช้จ่าย และยังสะดวกสบายในการสื่อสาร ทำให้การจัดการมีระเบียบมากขึ้น อีกทั้งยังทำให้ได้เปรียบคู่แข่งในการโฆษณาหรือขายสินค้าได้ถ้ามีการบริโภคข้อมูลข่าวสารอย่างถูกวิธี ด้วยเหตุนี้ส่งผลให้ข้อมูล ข่าวสารองค์ความรู้มีมากขึ้นทวีคูณ การจะเข้าถึงข้อมูลที่มากมายนั้นทำได้ยากจึงจำเป็นต้องมีตัวช่วยซึ่งเรียกว่า การค้นหา (search engine) การค้นหาคือการที่ผู้ใช้ใส่คำที่เกี่ยวข้องกับข้อมูลที่ต้องการลงไปให้ระบบรู้จัก แล้วระบบจะทำการค้นหาข้อมูลนั้นและแสดงออกมาแก่ผู้ใช้ โดยการค้นหาทำได้ทั้งบนเว็บเพจ ผ่านเว็บที่มีการเตรียมฐานข้อมูลเพื่อให้บริการได้แก่ google.co.th หรือ yahoo.com เป็นต้น และค้นหาบนไฟล์ข้อมูล ผ่านเครื่องมือต่าง ๆ ที่มีคนสร้างเอาไว้

การค้นหามีได้หลายชนิด แต่ชนิดที่ใช้ในโครงงานนี้คือการทำ Full-Text Search วิธีการค้นหาแบบ full-text คือการอ่านข้อมูลทั้งหมดแล้วมาสร้างเป็นดัชนีของข้อมูล เสร็จแล้วนำดัชนีของข้อมูลที่ได้นี้ไปใช้ในการค้นหา ดังนั้นสิ่งแรกที่ต้องทำคือสร้างดัชนีของข้อมูลโดยการอ่านข้อมูลจากไฟล์หรือฐานข้อมูลแล้วนำมาตัดคำแต่ละคำมาสร้างเป็นดัชนีเพื่อชี้กลับไปยังไฟล์ที่มีคำนี้อยู่และอาจจะมีค่า rank ของแต่ละคำเพื่อบอกถึงความบ่อยที่เจอคำนี้

จะเห็นว่าเครื่องมือที่ใช้ในการค้นหาแต่ละอันได้ข้อมูลเหมือนกันบ้าง ไม่เหมือนกันบ้าง ดังนั้นเพื่อพิสูจน์ให้ได้ว่าอันไหนมีข้อดีต่างกันอย่างไร อันไหนเหมาะกับการใช้งานในสถานการณ์อย่างไรจึงจำเป็นต้องทำการวิเคราะห์เปรียบเทียบเครื่องมือที่ใช้ในการทำดัชนีข้อมูล ในโครงงานนี้หยิบเครื่องมือมาทำการวิเคราะห์เปรียบเทียบสองอันได้แก่ Apache Lecene และ Sphinx ซึ่งสองชนิดที่หยิบมาใช้นี้เป็นรู้จักกันอย่างแพร่หลาย

1.2 วัตถุประสงค์ของการศึกษา

เพื่อเปรียบเทียบเครื่องมือที่ใช้ในการค้นหาด้านเวลาที่ใช้ และความถูกต้อง เพื่อที่จะนำเครื่องมือนั้น ๆ ไปใช้ได้ถูกงานและตามความต้องการได้

1.3 ขอบเขตของการศึกษา

1. เอกสารที่ใช้เป็นข้อมูล เป็นไฟล์ ภาษาอังกฤษ ที่เป็น pdf , ฐานข้อมูล MySQL, xml
2. วัดความเร็วในการสร้าง ดัชนี และ การค้นหา
3. วัดการใช้หน่วยความจำ
4. วัดความถูกต้องโดยใช้ Precision และ Recall
5. ในเบื้องต้นเลือกเครื่องมือที่ใช้ในการค้นหามาทำการทดลองเพียงสองตัวที่มีชื่อเสียงเห็นที่นิยม คือ Lucene และ Sphink

1.4 ขั้นตอนของการศึกษา

1 ศึกษาความสามารถของเครื่องมือ คือ Lucene และ Sphink

2 ออกแบบการทดลอง

3 ทำการทดลอง โดยการสร้าง โปรแกรมเพื่อเรียกใช้เครื่องมือ และเพื่อวัดความสามารถตามที่ได้ออกแบบไว้ในข้อ 3

4 สรุปผลที่ได้จากการทดลองโดยแบ่งตามลักษณะงาน หรือตามจังหวะเวลาที่ทำงาน

5 เสนอแนะ ความเป็นไปได้ที่จะนำไปพัฒนาต่อยอดในรูปแบบอื่น ๆ

1.5 ประโยชน์ที่คาดว่าจะได้

1.ทราบถึงการทำงาน ในการสร้าง ดัชนีข้อมูล

2.ทราบว่า ควรจะใช้เครื่องมืออะไรเพื่อที่จะได้ ผลลัพธ์ที่มีความถูกต้องที่สุด

3.ทราบว่าควรจะใช้เครื่องมืออะไรในการที่ใช้เวลาในการจัดทำดัชนีน้อยที่สุด

4.ทราบถึงความสามารถของเครื่องมือที่ใช้ในการค้นหา

5.สามารถนำผลการทดลองไปต่อยอดในการทำงานอื่น ๆ

บทที่ 2
ทฤษฎีที่ใช้

2.1 รูปแบบคำค้น

คำค้นมี 2 แบบคือ ค้นคำศัพท์เพียงหนึ่งคำ (Single Terms) คือคำเพียงหนึ่งคำเช่น “test” หรือ “hello” และ วลี (Phrases) คือกลุ่มคำ ที่ถูกครอบด้วย Double quotes เช่น “hello dolly” คำศัพท์หลายๆ คำมาต่อกันจะมีได้หลายรูปแบบ ดังนี้

2.1.1 Boolean Operators

คือสัญลักษณ์พิเศษที่ช่วยเพิ่มประสิทธิภาพในการค้นหา จะประกอบไปด้วย AND”,”+”,”OR”, ”NOT” และ “-”

- **”OR”**

”OR” เป็น Operators สำหรับการเชื่อมคำ 2 คำ ที่อยู่ระหว่าง ”OR” โดย ”OR” จะทำการค้นหาคำที่เท่ากับคำ 2 คำที่เชื่อมกัน หรือ คำใดคำหนึ่งที่อยู่ระหว่าง ”OR” เป็นลักษณะของการ Union กันระหว่างสองคำ นอกจากนี้เรายังใช้สัญลักษณ์ “||” แทนได้ เช่น ต้องการค้นหา “Information Technology” หรือ “Information” ใช้คำสั่งดังนี้ " Information Technology " Information หรือ " Information Technology " OR Information

- **”AND”**

“AND” เป็นoperator สำหรับค้นหาเอกสารที่จะต้องมีคำ 2 คำที่ต้องการค้นหาในเอกสารที่ใดที่หนึ่งในเอกสาร เป็นลักษณะของการ Intersection กันระหว่างสองคำ ซึ่งโดยปรกติถ้าเราไม่ใส่ ”AND” ระบบสืบค้นจะทำการเชื่อมคำให้อัตโนมัติเหมือน กับว่าเราใส่ ”AND” เช่น

ถ้าเราต้องการค้นหาคำ " Information Technology " and " Technology Information "

- **“+”**

“+” เป็น operator สำหรับบังคับว่าคำที่อยู่หลัง สัญลักษณ์ “+” จะต้อง มีอยู่ที่ใดที่หนึ่งในเอกสารที่ต้องการค้นหา เช่นต้องการค้นหาเอกสารที่ประกอบด้วย “Information” และก็ประกอบด้วย “Information” ใช้ได้ดังนี้ + Information Technology

- **“NOT”**

“NOT” เป็น operator สำหรับตัดเอกสารที่ประกอบไปด้วยคำที่ตามหลัง NOT ออกไป ซึ่งเรา

สามารถใช้สัญลักษณ์ “!” แทนคำว่า NOT ได้ เช่น ต้องการค้นหาเอกสารที่ประกอบด้วย “Information Technology” แต่ไม่เอา “search engine” ใช้คำสั่งดังนี้ " Information Technology " NOT " search engine " operator ตัวนี้ไม่สามารถใช้กับคำหรือวลีเพียงหนึ่งคำได้ เช่น NOT " Information Technology "

- “-“

“-“ เป็น operator ยกเว้นการค้นหา คือจะตัดเอกสารที่ประกอบด้วยคำที่อยู่หลังสัญลักษณ์ “-” เช่นถ้าเราต้องการค้นหา “Information Technology” แต่ไม่เอา “search engine” ใช้คำสั่งดังนี้ " Information Technology " -" search engine "

การแก้ไขคำศัพท์ที่ใช้ในการค้นหาก็เพื่อเป็นการกำหนดขอบเขตการค้นหาของระบบสืบค้น โดยมีวิธีการค้นหาด้วยกันหลายวิธีดังนี้

2.1.2 Wildcard Searches

เป็นการค้นหาที่สนับสนุนการค้นหาอักขระตัวเดียวหรือหลาย ๆ อักขระรวมกันซึ่งก็คือคำศัพท์ (Terms) นั่นเอง (แต่จะไม่สนับสนุนการค้นหาแบบวลี) สัญลักษณ์ที่ใช้ในการค้นหาอักขระตัวเดียว คือ “?”สัญลักษณ์ที่ใช้ในการค้นหาอักขระหลายตัว คือ “*” การค้นหาอักขระตัวเดียว wildcard Searches จะค้นหาอักขระทุกๆ อักขระที่สามารถนำมาใส่แทน “?” แล้วอ่านเป็นคำได้ เช่น “text” หรือ “test” วิธีการใช้ดังนี้ te?t การค้นหาอักขระหลายตัว wildcard Searches จะค้นหาอักขระตั้งแต่ 0 ตัวหรือมากกว่า เช่นต้องการค้นหาคำว่า test,tests หรือ tester ใช้วิธีการได้ดังนี้ test* นอกจากนี้เรายังสามารถใช้ * เพื่อแทนการค้นหาอักขระตัวเดียวได้โดย ใช้วิธีการดังนี้ te*t

หมายเหตุ: เราไม่สามารถวางสัญลักษณ์ “?” หรือ “*” ไว้หน้าสุดของคำที่ต้องการจะค้นหาได้

2.1.3 Fuzzy Searches

เป็นการค้นหาที่อยู่บนพื้นฐานของ Levenshtein Distance หรือ Edit Distance algorithm ซึ่งจะใช้สัญลักษณ์ “~” ใส่ลงไปที่ท้ายสุดของคำศัพท์ที่ต้องการค้นหา เช่น ถ้าต้องการค้นหาคำที่สะกดคำคล้ายๆกับคำว่า “roam” วิธีการใช้ ดังนี้ roam~ ผลที่ได้จากการค้นหาคำนี้ก็จะได้คำว่า foam และ roam แต่ในระบบสืบค้นนี้ได้มีการเพิ่มคุณพารามิเตอร์เข้ามาอีกเพื่อให้ผู้ใช้ ได้สามารถความใกล้เคียงของคำได้มากขึ้น โดยค่าที่ใส่เข้าไปมีค่าอยู่ระหว่าง 0-1 โดยค่าที่ใกล้เคียง 1 มากที่สุดจะเป็นคำที่คล้ายกับคำที่ต้องการค้นหามากทุกที่สุด เช่น roam0.8 แต่โดยทั่วไปถ้าหากไม่ระบุพารามิเตอร์ ก็จะมีค่าเป็น 0.5 ให้อยู่แล้ว

2.1.4 Proximity Searches

เป็นการค้นหาคำที่อยู่ในช่วงที่กำหนด โดยใช้สัญลักษณ์ “~” ต่อท้ายวลีที่ต้องการค้นหา เช่น ต้องการค้นหา “apeche” และ “jakarta” ที่จะต้องมีคำพวกนี้ 10 คำของแต่ละเอกสาร วิธีใช้ดังนี้ “jakarta apache”~10

2.1.5 Range Searches

เป็นการค้นหาโดยค่าของคำในเอกสารที่เราต้องการค้นหา ซึ่งค่าของคำนั้นจะต้องอยู่ระหว่างค่าที่เรากำหนด โดยจะเป็นการกำหนดแบบครอบคลุม (Inclusive) หรือแบบเฉพาะเจาะจง (exclusive) ก็ ได้ เช่น mod_date:[20020101 TO 20030101] ระบบจะทำการค้นหา mod_date ในเอกสารที่มีค่าอยู่ระหว่าง 20020101 และ 20030101 แต่โดยรวมแล้ว Range Searches ไม่ได้มีไว้สำหรับการหาแบบ date ดังนั้นเราจึงควรใช้วิธีการนี้ในการค้นหาแบบที่ไม่ใช่ date ดังนี้ title:{Aida TO Carmen} ระบบค้นหาค่าของ title ทุกๆคำที่อยู่ในเอกสาร ที่ค่าของ title อยู่ระหว่าง Aida และ Carman แต่จะไม่รวม Aida และ Carman โดยที่รูปแบบการกำหนดแบบ inclusive จะใช้สัญลักษณ์แทนโดย “[]” และ exclusive จะใช้สัญลักษณ์แทนโดย “{}”

2.1.6 Boosting a Term

เป็นตัวกำหนดระดับความสัมพันธ์ของการค้นหาคำศัพท์ในเอกสารที่ต้องการ โดยจะใช้สัญลักษณ์ “^” แล้วตามด้วยตัวเลข (the boost factor) ที่ท้ายคำศัพท์ที่เราต้องการค้นหา โดยที่ระดับตัวเลขที่มากความ สัมพันธ์ของคำศัพท์ก็จะมากตามไปด้วย เช่นถ้าคุณต้องการค้นหา jakarta apache แล้วเราต้องการคำศัพท์ jakarta ให้มีความสัมพันธ์กันมากขึ้น ก็โดยใช้สัญลักษณ์ “^” ซึ่งก็จะอยู่ในรูปแบบดังนี้ jakarta^4 apache นอกจากนี้เรายัง สามารถใช้ Boosting a Term กับวลีได้ด้วย โดยมีลักษณะดังนี้ "jakarta apache"^4 "Apache Lucene" แต่ถ้าเราไม่กำหนดปรกติ Boosting a Term จะมีค่าเท่ากับ 1 โดยที่ค่า the boost factor สามารถน้อยกว่า 1 ได้ แต่ไม่สามารถเป็นจำนวนลบได้

2.1.7 Grouping

จะใช้ () เป็นตัว Group เพื่อเป็นการแยกย่อยการค้นหา ซึ่งจะมีประโยชน์อย่างมากเมื่อเราต้องการควบคุมการใช้ operator หลายๆ ตัวในการค้นหา เช่น ต้องการค้นหา "jakarta" or "apache" และ “website” คำสั่งดังนี้ (jakarta OR apache) AND website เพื่อเป็นการจำกัดความสับสนและให้แน่ใจว่าเอกสารที่เราต้องการค้นหานั้น มีคำว่า “website” อยู่แน่นอนและนอกจากนี้จะต้องประกอบไปด้วย jakarta หรือ apache ในเอกสาร

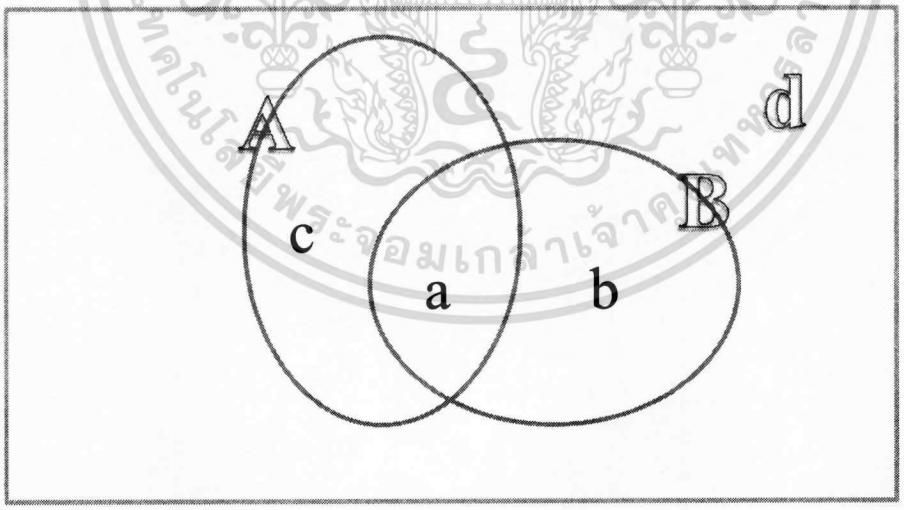
2.1.8 Escaping Special Characters

สัญลักษณ์พิเศษที่ระบบค้นหาจะไม่สามารถไปใช้เป็นคำค้นหาได้ มีดังนี้ + - && || ! ( ) { } [ ] ^ " ~ * ? : \ แต่ถ้าผู้ใช้ต้องการใช้สัญลักษณ์พิเศษดังกล่าวเพื่อการค้นหาจะต้องใส่สัญลักษณ์ "\" นำหน้าตลอด เช่น (1+1):2 ต้องเขียนดังนี้ \(1\+1\)\:2

2.2 การวัดประสิทธิภาพในการค้นหา

วัดประสิทธิภาพในการค้นหาด้วย Precision คือการวัด ความแม่นยำในการค้นคืน และ Recall คือความสามารถในการค้นคืนข้อมูล

การประเมินประสิทธิภาพความสามารถในการค้นหานั้นนิยมแสดงออกมาเป็นกราฟ Precision และ Recall หรือเรียกว่า กราฟ PR โดยค่า Recall เท่ากับ 1 หมายถึงระบบสามารถค้นหาข้อมูลที่เกี่ยวข้องกับความต้องการผู้ใช้ทั้งหมดออกมาได้ เช่นเดียวกับ Precision ที่มีค่าสูงหมายถึงระบบสามารถค้นหาข้อมูลออกมาได้โดยมีข้อมูลที่ไม่เกี่ยวข้องปะปนอยู่น้อยที่สุด ดังนั้นค่า Precision และ ค่า Recall จึงมักจะนำมาพิจารณาร่วมกัน เช่น พิจารณาค่า Precision ที่ Recall เท่ากับ 0.5 เป็นต้น สามารถกล่าวได้ว่าระบบค้นหามีประสิทธิภาพดีถ้ามีค่า Precision สูงด้วยค่า Recall ที่เท่ากัน



รูปที่ 2.1 อธิบายความหมายของ Precision และ Recall

A คือเซตของข้อมูลที่ต้องการทั้งหมด ที่อยู่ในข้อมูลต้นทาง

B คือเซตของข้อมูลที่ได้จากการค้นหา

d คือเซตของข้อมูลที่ไม่ต้องการ และระบบไม่สามารถค้นหาได้

a คือข้อมูลที่ต้องการที่ระบบสามารถค้นหาออกมาได้

c คือข้อมูลที่ต้องการแต่ระบบไม่สามารถค้นหาออกมาได้

b คือข้อมูลที่ไม่ต้องการแต่ระบบค้นหาออกมาได้

สามารถให้คำนิยามของค่า Precision และ Recall ได้ดังนี้

$$\text{Recall} = \frac{\text{จำนวนของเอกสารที่เกี่ยวข้องและถูกดึงออกมา}}{\text{จำนวนของเอกสารที่เกี่ยวข้องทั้งหมด}} = P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{a}{a+c}$$

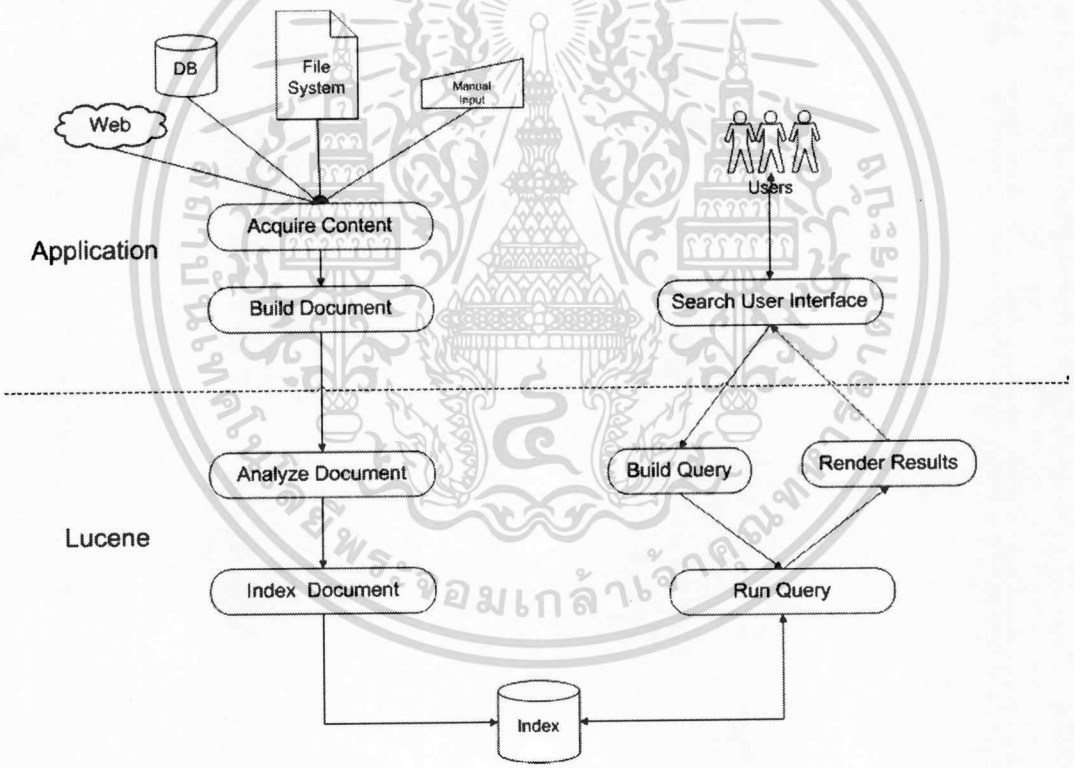
$$\text{Precision} = \frac{\text{จำนวนของเอกสารที่เกี่ยวข้องและถูกดึงออกมา}}{\text{จำนวนของเอกสารที่ถูกดึงออกมาทั้งหมด}} = P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{a}{a+b}$$

บทที่ 3
เครื่องมือที่ใช้ในการทดลอง

ในการทดลองนี้มีเครื่องมือที่ใช้ในการทดลองสองชนิดคือ Apache Lucene และ Sphinx ซึ่งสองชนิดใช้ในการค้นหาเหมือนกัน โดย ข้อมูลของเครื่องมือทั้งสองชนิดอธิบายได้ดังต่อไปนี้

3.1 APACHE LUCENE

การทำSearch Engine ในโครงงานนี้ได้มีการใช้ API ซึ่งทำมาจาก Java ชื่อว่า Lucene เป็นไลบรารี่ที่ใช้ในการค้นคืนข้อมูล ใช้ในการสร้างดัชนีและทำการค้นหา อยู่ในตระกูล Apache Jakarta เราสามารถนำมาประยุกต์ใช้ได้หลายแบบ Lucene มีส่วนประกอบต่าง ๆ ดังรูปด้านล่าง



รูปที่ 3.1 แสดงการทำงานของ Lucene

การสร้างดัชนีก็เพื่อที่จะค้นหาข้อมูลในหลาย ๆไฟล์ได้ง่ายและรวดเร็ว โดยการแปลงรูปแบบในไฟล์ที่มีคำและวลีอยู่ภายในไปเป็นรูปแบบที่ทำให้สามารถค้นหาได้เร็ว เราเรียกกระบวนการนี้ว่า Indexing และเรียกผลลัพธ์ของกระบวนการนี้ว่า Index

Acquire content

เป็นกระบวนการแรกในการสร้างดัชนีโดยจะบอกถึงขอบเขตของข้อมูล หรือการจัดการข้อมูล ในกระบวนการนี้ Lucene ไม่มีตัวจัดการ จึงต้องมีการจัดการใน แอปพลิเคชั่นแทนหรืออาจจะหาตัวไต่เว็บมาใช้ในกรณีที่ข้อมูลเข้าเป็นข้อมูลจากเว็บไซต์ต่าง ๆ ข้อมูลที่สามารถใช้ได้ได้แก่ เว็บ , ไฟล์ , ฐานข้อมูล หรือการกรอกข้อมูลเข้าไป จากกระบวนการนี้ได้มีผู้นำ Lucene ไปต่อยอดเพื่อให้เหมาะกับสายงานด้านอื่นเช่น Solr , Nutch , Gurd เป็นต้น

Build Document

เป็นการนำข้อมูลในไฟล์ไปจัดเอกสารเพื่อแยกประเภท โดยอาจจะแบ่งเป็นฟิลด์เช่น หัวข้อ , เนื้อหา , ผู้เขียน หรือชื่อเอกสารเป็นต้น ส่วนใหญ่ข้อมูลจะมีสองชนิดคือแบบ ASCII กับ binary ใน API ของ Lucene ยังไม่สามารถรับข้อมูลจาก ฐานข้อมูลได้ แต่ได้มีผู้นำไปต่อยอดเพื่อให้รองรับข้อมูลจากฐานข้อมูลโดยกำหนด ขอบเขตของข้อมูล และการจัดการเอกสารให้รับข้อมูลประเภทฐานข้อมูลได้แล้ว

Analyze Document

ขั้นตอนนี้ไม่ใช่การค้นหาข้อมูลโดยตรงแต่จะมีการตัดคำเป็นชุด ๆ เรียกว่า Token อาจจะมีการแบ่งชุดตามตัวสะกด หรือแบ่งตามความหมายที่ใกล้เคียงกันโดยผู้ใช้มีการกำหนดคำศัพท์ที่มีความหมายเหมือนกันเป็นชุด ๆเตรียมไว้

Index Document

เอกสารต่าง ๆได้ถูกสร้างดัชนีขึ้นมาเพื่ออ้างไปยังเอกสารต้นทางโดยสามารถเพิ่ม ลบ หรือแก้ไขได้ แบบ เรียลไทม์และไม่เรียลไทม์

ในส่วนของการค้นหาเป็นการค้นหาในดัชนีที่ถูกสร้างไว้เพื่อที่จะอ้างถึงเอกสาร ประสิทธิภาพของการค้นหาข้อมูลบอกได้ด้วยค่า Precision และ Recall การทำงานเริ่มที่

Search User Interface

เป็นหน้าที่ไว้ให้ผู้ใช้งานกรอกคำร้องขอดูข้อมูล และยังมีไว้แสดงผลของการค้นหาด้วยว่ามีอะไรบ้าง

Build Query

ขั้นตอนนี้มีการแปลงข้อความที่ผู้ใช้กรอกเข้าไป โดยในการค้นหามีการค้นหาได้หลายรูปแบบตามที่กล่าวไปในข้อ 2.1

Search Query

เป็นกระบวนการที่นำคำร้องขอไปค้นในดัชนีเพื่อที่จะชี้ไปยังข้อมูลต้นทางมี สามรูปแบบ ได้แก่

- **Pure Boolean model** เป็นการจับคู่กันระหว่างคำศัพท์ดัชนีและคำค้น
- **Vector space model** แทนเอกสารและคำค้นในรูปเวกเตอร์ โดยกำหนดค่าน้ำหนักของคำด้วยความถี่ของคำที่ปรากฏในเอกสาร
- **Probabilistic model** จัดลำดับเอกสารในมวลทรัพยากรสารสนเทศตามความน่าจะเป็นด้านความเข้าเรื่องของแต่ละเอกสารกับข้อคำถาม โดยเรียงลำดับจากมากไปหาน้อย

ที่นิยมในปัจจุบันมีสองชนิดคือ แบบ Boolean และ vector

Render Result

ทำหน้าที่ในการจับคู่ผลกับคำค้นที่ส่งเข้ามาเพื่อที่จะนำไปแสดงใน หน้าจอ

Lucene รองรับการนำข้อมูลมาพิจารณาได้หลาย รูปแบบ ทั้ง .doc ,.txt ,.pdf ,.html ผู้ใช้สามารถ ใส่ข้อมูลเป็น ไฟล์ ,เป็นชื่อเว็บ หรือ ใส่เป็นข้อมูลได้เลย เพื่อที่จะนำไปสร้างเป็น ดัชนี และเมื่อต้องการดูข้อมูลก้อสามารถรับคิวรี่ มาค้นหาใน ดัชนี และนำข้อมูลที่ต้องการดูมาแสดงได้ถูกต้อง

การทำ ดัชนีของ Lucene นั้น มี คลาส ที่ถูกใช้งานหนึ่งอันเรียกว่า Indexer ตัว Indexer นี้จะรับข้อมูลอยู่สองอย่างคือ ที่อยู่ของเว็บที่ต้องการเก็บข้อมูล และ ที่อยู่ของไฟล์ดัชนีที่จะถูกสร้าง ผลที่ได้คือ ไฟล์ดัชนีที่เป็น รูปแบบที่ตัวมันเองอ่านออก

การสร้าง ดัชนี ของ Lucene

ประกอบไปด้วย

1 IndexWriter
2 Directory
3 Analyzer
4 Document
5 Field

ทำได้โดย เริ่มแรก class Indexer จะถูกเรียก พร้อมกับรับค่า ไดเร็กทอรี่ที่ต้องการเก็บดัชนี และ ไดเร็กทอรี่ที่เก็บไฟล์ของข้อมูลที่ต้องการค้นหา เก็บใส่ใน ตัวแปร

```
File indexDir = new File (args[0]);
File dataDir = new File (args[1]);
```

จะมีการสร้างไฟล์ดัชนี

```
IndexWriter writer = new IndexWriter(indexDir,  new  StandardAnalyzer(),true);
Writer.setUseCompoundFile(false);
```

หลังจากนั้นจะมีการวนลูปอ่านข้อมูลและพิจารณาสร้างเป็นดัชนี และเก็บใส่ไฟล์ตามไดเร็กทอรี่ที่กำหนด

```
For ( int i = 0; i< files.length; i++) {
        File f = files(i);
        If( f.isDirectory()){
          indexDirectory( writer , f );
        } else if ( f.getName () .endsWith(".txt")){
          indexFile(writer , f );
        }
}
```

พิจารณาค่าดัชนีและเก็บลงไฟล์

```
        Document  doc  =  new  Document ();
        doc.add( Field.Text("contents", new FileReader ( f ))) ;
        doc.add( Field.Keyword ( "filename" , f.getCanonicalPath())) ;
        writer.addDocument ( doc );
```

ถ้าต้องการให้ใช้ Analyzer ให้ใช้เป็น addDocument(doc,Analyzer)

การค้นหาข้อมูล ของ Lucene

ประกอบไปด้วย

1 IndexSearcher

2 Term

3 Query

4 TermQuery

5 TopDocs

ทำได้โดย การค้นหาข้อมูลในดัชนีของ Lucene มี คลาส ที่เกี่ยวข้องคือ Searcher ตัว Searcher นี้จะรับข้อมูลสองอย่างคือ ที่อยู่ของดัชนี และ ความต้องการร้องขอดูข้อมูลหรือคิวรี่ ที่ผู้ใช้ต้องการจะดู

เริ่มแรกคลาส Searcher จะถูกเรียก พร้อมกับรับค่า ไดเร็กทอรี่ของดัชนี และ คิวรี่ข้อมูลที่ต้องการค้นหา

```
File indexDir = new File(args[0]);
String q = args[1];
```

เปิดดัชนี และ วางคิวรี่เพื่อค้นหา

```
Directory fsDir = FSDirectory.getDirectory( indexDir , false);
IndexSearcher is = new IndexSearcher( fsDir ) ;
Query query = QueryParser.parse ( q, "contents" ,
new StandardAnalyzer());
```

ค้นหา โดยใช้คำสั่ง

```
Hits hits = is.search(query);
```

วนลูปเพื่อที่จะจับคู่กับเอกสาร

```
For( int i = 0; i< hits.length () ; i++) {
    Document  doc = hits.doc(i) ;
    System.out.println(doc.get( " filename"));
}
```

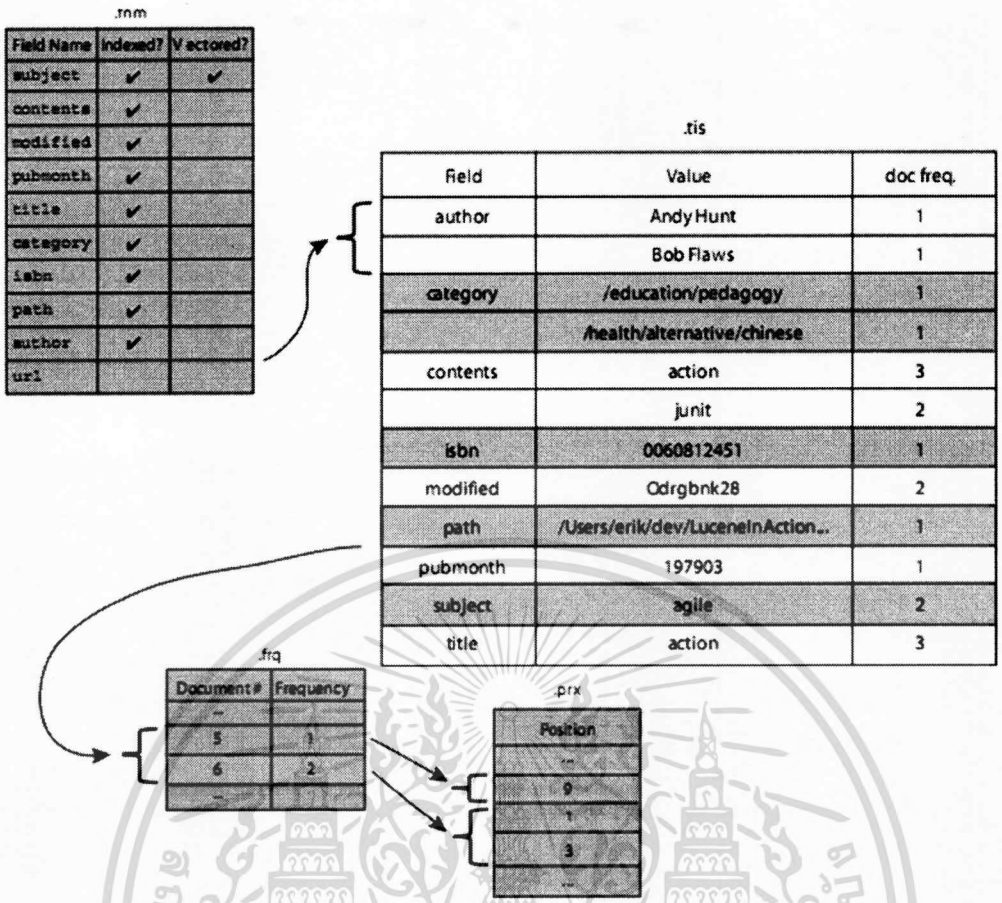
โครงสร้างของดัชนีที่ได้จาก Lucene มีสองรูปแบบ คือ แบบ multifile และ แบบcompound

แบบ mulfile

จะมีการอ่านค่าในไฟล์เพื่อนำค่าในไฟล์ไปสร้าง index ได้ไฟล์ต่าง ๆ ตามรูปที่ 3.2

_r.fdt	FDT File	14 KB
_r.fdx	FDX File	2 KB
_r.fnm	FNM File	1 KB
_r.frq	FRQ File	4 KB
_r.prx	PRX File	4 KB
_r.tii	TII File	1 KB
_r.tis	TIS File	6 KB
segments.gen	GEN File	1 KB
segments_j	File	1 KB

รูปที่ 3.2 แสดงไฟล์ที่ได้จากการรัน multifile index



รูปที่ 3.3 แสดงโครงสร้างของ index ที่ได้จาก Lucene

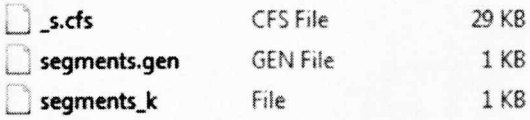
จากรูปที่ 3.3 จะเห็นว่าใน .fnm จะมีการกำหนด field เอาไว้ และเมื่ออ่านข้อมูลเข้าไปจะไปอยู่ใน .tis จะมี field ที่ได้กำหนดไว้ และยังมีความถี่ที่เจอเพื่อเชื่อมไปยังเอกสารต้นทางใน .frq และ .prx การสร้าง รูปแบบนี้ทำให้สะดวกในการที่จะเพื่อเอกสารเข้าไปได้

การเรียกใช้งาน ให้สร้างแบบ multifile

```
IndexWriter writer = new IndexWriter(indexDir,
    New StandardAnalyzer(Version.LUCENE_30) ,
    True, Indexwriter.MaxFieldLength.UNLIMITED);
Writer.setUseCompoundFile(false);
```

แบบ Compound

เมื่อสั่งให้สร้าง compound จะเหลือไฟล์ดัชนีเพียงแค่ 3 ไฟล์ จากข้อมูลเข้าเหมือนกันกับแบบ multifile ดังรูปที่ 3.4 การสร้างรูปแบบนี้ทำให้การอ่านindex เร็วขึ้นเนื่องจากเหือไฟล์ที่เก็บเพียงแค่สามไฟล์แต่จะยากในการเพื่อข้อมูลเอกสาร



รูปที่ 3.4 แสดงไฟล์ที่ได้จากการรัน compound index

การเรียกใช้งาน ให้สร้างแบบ compound

```
IndexWriter writer = new IndexWriter(indexDir,
   New StandardAnalyzer(Version.LUCENE_30) ,
   True, Indexwriter.MaxFieldLength.UNLIMITED);
Writer.setUseCompoundFile(true);
```

3.2 SPHINX

Sphinx เป็นเครื่องมือที่ใช้ในการค้นหาแบบ Full – Text ชนิดหนึ่ง สามารถใช้งานได้กับระบบปฏิบัติการ Linux , Windows 2000, XP ,Mac OS โดยเข้าถึงได้สามทาง คือ SphinxAPI (ใช้เพื่อติดต่อกับภาษาอื่น ๆ เช่น PHP, Perl, Ruby, and Java), SphinxQL (ใช้เพื่อติดต่อกับ mysql) ,SphinxSE (ใช้เพื่อติดต่อกับ SQL server) ข้อมูลที่ใช้สามารถโหลดเข้าไปเพื่อนำไปสร้าง index ได้สองทางคือ ผ่านทางฐานข้อมูล และผ่านทาง XML

ความสามารถของ Sphinx ที่ได้ระบุไว้คือ

- การสร้างดัชนีและการค้นหาสามารถทำได้เร็ว ที่ความเร็วที่ได้บอกเอาไว้คือ ใช้สร้างดัชนี 10-15 MB/วินาที ใช้ค้นหาที่ความเร็ว 150-250 คำค้น /วินาที ที่ ข้อมูล 1.2 GB
- มีเครื่องมือที่ใช้ค้นหาขั้นสูง (ยืดหยุ่น ในการตัดคำ , ใช้ภาษา การค้นหาได้, มีการให้ค่าคำค้น);
- การใช้งานค้นหาเหมือนกันภาษา SQL คือมีการใช้ SELECT , WHERE, ORDER BY, GROUP BY และอื่น ๆ)
- สะดวกในการใช้งานกับ ฐานข้อมูลต่าง ๆ
- มีความเร็วในการทำ ดัชนีที่ 10 MB/วินาที บน CPUs)
- มีความเร็วในการค้นหาที่ เฉลี่ย การค้นหา 0.1 วินาทีได้ 2-4 GB
- high scalability (up to 100 GB of text, up to 100 M documents on a single CPU)
- รองรับการทำ distributed searching
- รองรับการทำ Full text บน MySQL

- รองรับการค้นวลี
- รองรับเอกสารที่เป็นกลุ่ม ๆ
- รองรับ stopwords
- รองรอบการค้นหาในโหมดที่แตกต่างกัน
- สามารถสร้าง XML interface เพื่อนำไปต่อยอดพัฒนาอย่างอื่น
- ใช้ภาษา PHP ในการพัฒนา

เครื่องมือที่ใช้ใน Sphinx มีดังนี้

- Indexer ใช้ในการสร้าง ดัชนีข้อมูล
- search ใช้ในดารค้นหาข้อมูล
- searchd ใช้เพื่อให้โปรแกรมอื่นภายนอกสามารถที่จะเห็นได้
- sphinxapi ใช้เพื่อเป็น ไลบรารี่สำหรับการค้นหาจากเว็บด้วยภาษาต่าง ๆ เช่น PHP, Python, Perl, Ruby เป็นต้น
- spelldump
- indextool

Indexing

การสร้างดัชนีของ Sphinx สามารถโหลดข้อมูลเข้าได้ สองทางคือจากฐานข้อมูล และจาก XML

จากฐานข้อมูล ต้องแก้ข้อมูลใน sphinx.conf เป็น

```
source src1
{
type            = mysql
sql_host        = localhost
sql_user        = root
sql_pass        = root
sql_port        = 3306
sql_db          = source1
sql_query       = select id,id1, unix_timestamp(date_added) as date_added ,content
from documents
sql_query_info  = select * from documents where id=$id
sql_attr_timestamp  = date_added
sql_ranged_throttle  = 0
```

```
}
index source1
{
    source                  = src1
    path                    = c:/data/test1
    docinfo                 = extern
    charset_type            = sbcs
}
```

จาก XML ต้องแก้ข้อมูลใน sphinx.conf เป็น

```
source scr1
{
    type            = xmlpipe2
    xmlpipe_command  = c:/sphinx/makeindex.php
    xmlpipe_field   = title     // กำหนด field ที่ใช้ในการค้นหา
    xmlpipe_field   = content
    xmlpipe_attr_uint = author_id
    xmlpipe_attr_timestamp = publish_date
    xmlpipe_attr_multi = category_id
}
index posts
{
    source                  = blog
    path                    = c:/data/scr1-xmlpipe2
    docinfo                 = extern
    charset_type            = utf-8
}
```

ถ้าต้องการให้มีการกำหนดขอบเขตการใช้เมมโมรี่ในการสร้างดัชนีให้กำหนด mem_limit **ลงไปในไฟล์** .conf **ตามด้านล่าง**

```
indexer
{
    mem_limit               = 32M
}
```

บทที่ 4

การออกแบบและวิธีทำการทดลอง

การเปรียบเทียบความสามารถและคุณสมบัติของเครื่องมือที่ใช้ในการทำดัชนีของข้อมูลนั้น มีด้วยกันหลายด้าน ส่วนที่นำมาใช้ในโครงการนี้ได้แก่

4.1 หลักเกณฑ์ที่ใช้ในการวัด คือ

1. วัดเวลาที่ใช้ในการสร้างดัชนี
2. วัดเวลาที่ใช้ในการค้นหา
3. วัดความถูกต้องในการค้นหา

4.2 เอกสารที่ใช้ในการทดลอง

ไฟล์ที่นำมาใช้ในการค้นหาเป็นบทความโครงงานที่ได้โหลดมาจากเว็บ www.scienccedirect.com จำนวน 100 ไฟล์ เนื้อหาเกี่ยวกับบทความคอมพิวเตอร์ ชนิดของไฟล์คือ PDF โดยได้ตั้งชื่อไฟล์ เป็น Sdarticle1- Sdarticle 100 ตามที่ได้แสดงไว้ในส่วนของภาคผนวก โดยใน 100 ไฟล์มีขนาดตั้งแต่ 72 KB ไปจดถึง2,239 KB รวมแล้ว ทั้งหมด ขนาด 60.7 MB

การสร้างดัชนี

ขั้นตอนทดสอบการทำงานของ Sphinx ในการสร้างดัชนี

1 นำเอกสารที่เตรียมมา จำนวน 100 ไฟล์ จัดรูปแบบให้อยู่ในรูป XML นำไปวางไว้ในโฟลดอร์ที่กำหนดไว้

2 เรียกใช้โปรแกรม Sphinx เพื่อให้ไปอ่านไฟล์ในโฟลเดอร์ที่กำหนดเพื่อสร้างดัชนีจากเอกสารไว้ในโฟลเดอร์ที่กำหนด

3 ดูเวลาที่ใช้ในการสร้างดัชนีของ Sphinx

ขั้นตอนทดสอบการทำงานของ Lucene ในการสร้างดัชนี

1. นำเอกสารที่เตรียมมา จำนวน 100 ไฟล์ ในรูป PDF นำไปวางไว้ในโฟลเดอร์ที่กำหนดไว้

2. เรียกใช้โปรแกรม Lucene เพื่อให้ไปอ่านไฟล์ในโฟลเดอร์ที่กำหนดเพื่อสร้างดัชนีจากเอกสารไว้ในโฟลเดอร์ที่กำหนด
3. ดูเวลาที่ใช้ในการสร้างดัชนีของ Lucene

ขั้นตอนเปรียบเทียบการทำงานของ Lucene และ Sphinx ในการสร้างดัชนี

1. นำเอกสารที่เตรียมมา จำนวน 100 ไฟล์ ใส่หัวข้อของเอกสารนั้น ๆ ลงไปในฐานข้อมูล MySQL ที่มี โครงสร้างของฐานข้อมูลดังนี้

file	date_added	content	id
Sdarticle1	2011-05-24 00:00:00	Instantiating abstract argumentation with classical Postulates and properties	1
Sdarticle2	2011-05-24 00:00:00	Efficient solutions to factored MDPs with imprecise transition probabilities	2
Sdarticle3	2011-05-24 00:00:00	Local closed world reasoning with description logics under the ell-founded semantics	3
Sdarticle4	2011-05-24 00:00:00	Hybrid trancability of valued constraint problems	4
Sdarticle5	2011-05-24 00:00:00	Local Search with edge weighting and configuration checking heuristics for minimum vertex cover	5
Sdarticle6	2011-05-24 00:00:00	Inconsistent heuristics in theory and practice	6
Sdarticle7	2011-05-24 00:00:00	Learning qualitative models from numerical data	7
Sdarticle8	2011-05-24 00:00:00	Voting almost maximizes social welfare despite limited communication	8
Sdarticle9	2011-05-24 00:00:00	Towards a model of musical interaction and communication	9
Sdarticle10	2011-05-24 00:00:00	Dedekind categories with cutoff operators	10
Sdarticle11	2011-05-24 00:00:00	SIP Security and the IMS core	11
Sdarticle12	2011-05-24 00:00:00	Ambiguous representation as fuzzy relations between sets	12
Sdarticle13	2011-05-24 00:00:00	L-topological spaces as spaces of points	13
Sdarticle14	2011-05-24 00:00:00	Quantitative domains via fuzzy sets: Part II : Fuzzy Scott topology on fuzzy directed-complete posets	14
Sdarticle15	2011-05-24 00:00:00	Fuzzy algebras as a framework for fuzzy topology	15
Sdarticle16	2011-05-24 00:00:00	Selected papers of the Refinement Workshop Turku(2008)	16
Sdarticle17	2011-05-24 00:00:00	Simulation refinement for concurrency verification	17
Sdarticle18	2011-05-24 00:00:00	Completeness of fair ASM refinement	18
Sdarticle19	2011-05-24 00:00:00	A tactic language for refinement of state-rich concurrent specifications	19
Sdarticle20	2011-05-24 00:00:00	Signal processing	20
Sdarticle21	2011-05-24 00:00:00	Variable selection in linear regression: Several approaches based on normalized maximum likelihood	21
Sdarticle22	2011-05-24 00:00:00	Intracranial subdural hematoma as a cause of postoperative delirium and headache in cervical laminoplasty: A case re...	22

รูปที่ 4.1 ข้อมูลจำนวน 100 เรคคอร์ด

2. เรียกใช้โปรแกรม Sphinx เพื่อให้ไปอ่านข้อมูลในฐานข้อมูลที่กำหนดในไฟล์ config เพื่อสร้างดัชนีไว้ในโฟลเดอร์ที่กำหนด
3. เรียกโปรแกรม Lucene เพื่อให้ไปอ่านข้อมูลในฐานข้อมูลที่กำหนดในโปรแกรมเพื่อสร้างดัชนีไว้ในโฟลเดอร์ที่กำหนด
4. ดูเวลาที่ใช้ในการสร้างดัชนีของ Sphinx และ Lucene

การค้นหา

นำดัชนีที่ได้จากการสร้างด้านบนทั้งจาก Lucene และ Sphinx มาทำการค้นหาเอกสาร โดย

ขั้นตอนวิธีทดสอบ

ขั้นตอนทดสอบการทำงานของ Sphinx ในการค้นหา

1. นำดัชนีที่สร้างจากด้านบนว่าไว้ใน โฟลเดอร์ที่กำหนด
2. เรียกใช้โปรแกรมค้นหา Sphinx โดย ใส่คำค้น ดังต่อไปนี้

-Boolean Operators

AND 5 คำที่ไม่มี -Apache AND lucene , Information AND technology , search AND engine , Information AND Retrieve , Network AND Programming

5 คำที่มี - performance AND analysis , Local AND object , multiple AND speech , robust AND radio , tactic AND language

OR 5 คำที่ไม่มี – Apache OR lucene , Information OR technology , searchs OR engine , Information OR Retrieve , Network OR Programming

5 คำที่มี - performance OR analysis , Local OR object , multiple OR speech , robust OR radio , tactic OR language

NOT 5 คำที่ไม่มี – NOT multiple ,NOT tactic , NOT language , NOT robust , NOT analysis

5 คำที่มี – NOT apache , NOT lucene , NOT sphinx , NOT engine , NOT Information

- Wildcard Searches 5 คำที่ไม่มี – apach* , lucen* , sphin* , engine* , Informatio*

  5 คำที่มี - multip* , tacti* , langua* , robus* , analys*

- Fuzzy Searches 5 คำที่ไม่มี - apaches~ , lucenes~ , sphinks~ , engines~ , Informations~

  5 คำที่มี - - multiples~ , tactics~ , languages~ , robusts~ , analysises~

- Proximity Searches 5 คำที่ไม่มี apache^2 , lucene^2 , sphinx^2 , engine^2 , Information^2

  5 คำที่มี – Fuzzy^2 , Source^2 , dimention^2 , signal^2 , identification^2

- Range Searches 5 คำที่ไม่มี – id:{101 to 105} , id:{200 to 205} , file:{sdarticle200 to sdarticle205} , file:{ sdarticle101 to sdarticle105}, date_added:{1/05/2010 to 31/05/2010}

  5 คำทีมี – id:{1 to 5} , id:{1 to 5} , file:{ sdarticle1 to sdarticle5}, file:{ sdarticle50 to sdarticle55}, date_added:{1/05/2011 to 31/05/2011}

- Boosting a Term 5 คำที่ไม่มี – “Apache lucene”^2 , “Information technology”^2 , “search engine”^2 , “Information Retrieve”^2 , “Network Programming”^2

5 คำที่มี - “robust vedio”^1 , “baysian lasso”^1 ,”adaptive algorithm”^1,”fuzzy algebra”^1, “local search”^1

- Grouping 5 คำ ที่ไม่มี – (Apache OR lucene) AND sphinx , (Information OR technology) AND sphinx , (searchs OR engine) AND sphinx , (Information OR Retrieve) AND sphinx ,(Network OR Programming) AND sphinx

  5 คำที่มี - (Apache OR lucene) OR analysis, (Information OR technology) OR analysis, (searchs OR engine) OR analysis, (Information OR Retrieve) OR analysis,(Network OR Programming) OR analysis
- Escaping Special Characters 5 คำที่ไม่มี – Apache \- lucene, Information \-technology , search\- engine , Information \- Retrieve , Network \- Programming

  5 คำที่มี - Robust M\- periodogram , large\=scale , \ "high\-dollar\" , safety \– critical, encoded in\=mage

3. วัดค่า Precision และ Recall ที่ได้จากการค้นหา มาทำการหาค่าเฉลี่ยในการค้นหาแต่ละรูปแบบเพื่อดูถึงความถูกต้อง

ขั้นตอนทดสอบการทำงานของ Lucene ในการค้นหา

1. นำดัชนีที่สร้างจากด้านบนว่าไว้ใน โฟลเดอร์ที่กำหนด
2. เรียกใช้โปรแกรมค้นหา Lucene โดย ใส่คำค้น คำค้นที่ใช้เหมือนกันกับการทอลองด้านบน
3. วัดค่า Precision และ Recall ที่ได้จากการค้นหา มาทำการหาค่าเฉลี่ยในการค้นหาแต่ละรูปแบบเพื่อดูถึงความถูกต้อง

ขั้นตอนเปรียบเทียบการทำงานของ Lucene และ Sphinx ในการค้นหา

1. นำดัชนีที่สร้างจากด้านบนว่าไว้ใน โฟลเดอร์ที่กำหนด
2. เรียกใช้โปรแกรมค้นหา Sphinx และ Lucene โดย ใส่คำค้น คำค้นที่ใช้เหมือนกันกับการทอลองด้านบน
3. วัดค่า Precision และ Recall ที่ได้จากการค้นหา มาทำการหาค่าเฉลี่ยในการค้นหาแต่ละรูปแบบเพื่อดูถึงความถูกต้อง

4.3 วิธีการทดลอง

ลักษณะของเครื่องที่ทำการทดลองเป็น notebook CPU Intel core i3 2.27 GHz

Ram 4GB Harddisk 320 GB

ข้อมูลที่นำมาทดสอบ

รูปที่ 4.2 แสดงข้อมูลจำนวน 100 ไฟล์

การรัน Lucene เพื่อสร้าง ดัชนี

- นำข้อมูลไปไว้ที่ C: /LuceneData/
- รันโปรแกรม Luceneindex.java

  จากการสร้างดัชนี ผลที่ได้จะบอกจำนวนไฟล์ และเวลาที่ใช้ในการสร้างดัชนี

```
Indexing C:\LuceneData\sdarticle76.pdf
Indexing C:\LuceneData\sdarticle77.pdf
Indexing C:\LuceneData\sdarticle78.pdf
Indexing C:\LuceneData\sdarticle79.pdf
Indexing C:\LuceneData\sdarticle8.pdf
Indexing C:\LuceneData\sdarticle80.pdf
Indexing C:\LuceneData\sdarticle81.pdf
Indexing C:\LuceneData\sdarticle82.pdf
Indexing C:\LuceneData\sdarticle83.pdf
Indexing C:\LuceneData\sdarticle84.pdf
Indexing C:\LuceneData\sdarticle85.pdf
Indexing C:\LuceneData\sdarticle86.pdf
Indexing C:\LuceneData\sdarticle87.pdf
Indexing C:\LuceneData\sdarticle88.pdf
Indexing C:\LuceneData\sdarticle89.pdf
Indexing C:\LuceneData\sdarticle9.pdf
Indexing C:\LuceneData\sdarticle90.pdf
Indexing C:\LuceneData\sdarticle91.pdf
Indexing C:\LuceneData\sdarticle92.pdf
Indexing C:\LuceneData\sdarticle93.pdf
Indexing C:\LuceneData\sdarticle94.pdf
Indexing C:\LuceneData\sdarticle95.pdf
Indexing C:\LuceneData\sdarticle96.pdf
Indexing C:\LuceneData\sdarticle97.pdf
Indexing C:\LuceneData\sdarticle98.pdf
Indexing C:\LuceneData\sdarticle99.pdf
Indexing 100 files took 12189 milliseconds
```

รูปที่ 4.3 แสดงการรัน index ด้วย Lucene

จะมีการสร้างไฟล์ ดัชนีต่าง ๆ ใน C:/LuceneIndex โดยไฟล์ที่ได้จะมีรูปแบบตามรูปด้านล่าง

_0.cfs	CFS File	4,300 KB
_0.cfx	CFX File	6 KB
_1.cfs	CFS File	4,722 KB
_2.cfs	CFS File	4,393 KB
_3.cfs	CFS File	4,596 KB
_4.cfs	CFS File	4,621 KB
_5.cfs	CFS File	4,505 KB
_6.cfs	CFS File	425 KB
segments.gen	GEN File	1 KB
segments_2	File	2 KB

รูปที่ 4.4 แสดงไฟล์ดัชนีที่ได้จากการรัน index ด้วย Lucene

การรัน Sphinx เพื่อสร้างดัชนี

- นำข้อมูลไปไว้ที่ C:/Sphinxdata
- เปิด cmd แล้วเข้าไปที่ path ที่ sphinx อยู่ แล้วสั่งรัน indexing ด้วยคำสั่ง
- Indexer –config c:\sphinx/sphinx/sphinx.conf –all จะได้ผลตามรูปด้านล่าง

```
using config file 'c:\sphinx/sphinx/sphinx.conf'...
indexing index 'source1'...
collected 100 docs, 0.0 MB
sorted 0.0 Mhits, 100.0% done
total 100 docs, 9101 bytes
total 0.088 sec, 102729 bytes/sec, 1128.77 docs/sec
total 2 reads, 0.000 sec, 5.3 kb/call avg, 0.0 msec/call avg
total 7 writes, 0.000 sec, 3.9 kb/call avg, 0.1 msec/call avg
```

- **รูปที่ 4.5** แสดงการรัน index ด้วย Sphinx

- ภายในตัวของ indexer จะมีการคิดคำนวณเวลาที่ใช้ในการสร้าง index มาให้แสดงออกมา เมื่อรันเสร็จจากรูป ใช้เวลา ในการรัน 0.001 วินาที ผลที่ได้จากการรัน จะมีการสร้างไฟล์ต่าง ๆ มากมายเก็บอยู่ใน ไดเร็กทอรี่ที่ได้มีการกำหนดไว้ในไฟล์ Sphinx.conf
- จะมีการสร้างไฟล์ ดัชนีต่าง ๆ ใน C:/Sphinxindex โดยไฟล์ที่ได้จะมีรูปแบบตามรูปด้านล่าง

test1	25/4/2554 12:21	Flash Document
test1.spd	25/4/2554 12:21	SPD File
test1.sph	25/4/2554 12:21	SPH File
test1.spi	25/4/2554 12:21	SPI File
test1.spk	25/4/2554 12:21	SPK File
test1.spm	25/4/2554 12:21	SPM File
test1.spp	25/4/2554 12:21	SPP File
test1stemmed	25/4/2554 12:21	Flash Document
test1stemmed.spd	25/4/2554 12:21	SPD File
test1stemmed.sph	25/4/2554 12:21	SPH File
test1stemmed.spi	25/4/2554 12:21	SPI File
test1stemmed.spk	25/4/2554 12:21	SPK File
test1stemmed.spm	25/4/2554 12:21	SPM File
test1stemmed.spp	25/4/2554 12:21	SPP File

รูปที่ 4.6 แสดงไฟล์ Index ที่ถูกสร้างขึ้นมา

การรัน Lucene เพื่อค้นหา

- มีดัชนีใน C:/Lucenedata
- รันโปรแกรม LuceneSearch.java ใส่ค่าคำค้น จะได้ผลตามรูปด้านล่าง

```
Found 1 document(s) (in 15 milliseconds) that matched query 'tactic and language':
sdarticle19.pdf
```

รูปที่ 4.7 ผลการรัน Searcher

การรัน Sphinx เพื่อค้นหา

ในส่วนของการ ค้นหา ให้ไปที่ ที่ path ที่ sphinx อยู่ แล้วสั่งรันคำสั่ง

Search –config C:\sphinx/sphinx/sphinx.conf this

ในที่นี้ใช้ this เป็นคำสั่งที่ต้องการค้นหาผลที่ได้จะเป็นดังรูปด้านล่าง

```
C:\sphinx\sphinx\bin>Search --config C:\sphinx/sphinx/sphinx.conf affecting
Sphinx 0.9.9-release (r2117)
Copyright (c) 2001-2009, Andrew Aksyonoff

using config file 'C:\sphinx/sphinx/sphinx.conf'...
index 'source1': query 'affecting ': returned 11 matches of 11 total in 0.000 se
c

displaying matches:
1. document=30, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest30
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to be absorb
ed in a white-winged yacht going by.'Don't seem to look; she comes to be quiet a
nd enjoy herself. Pretend we don't see her, that's only civil,' answered Bess, a
ffecting to be absorbed in a white-winged yacht going by.
        id=30
2. document=31, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest31
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to be absorb
ed in a white-winged yacht going by.
        id=31
3. document=32, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest32
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to be absorb
ed in a white-winged yacht going
        id=32
4. document=33, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest33
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to be absorb
ed in a white-winged yacht
        id=33
        id1=doctest34
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to be absorb
ed in a white-winged
        id=34
6. document=35, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest35
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to be absorb
ed in a
        id=35
7. document=36, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest36
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to be absorb
ed in
        id=36
8. document=37, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest37
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to be absorb
ed
        id=37
9. document=38, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest38
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to be
        id=38
10. document=39, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest39
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting to
        id=39
11. document=40, weight=1, date_added=Mon May 09 00:00:00 2011
        id1=doctest40
        date_added=2011-05-09 00:00:00
        content=Don't seem to look; she comes to be quiet and enjoy herself. Pre
tend we don't see her, that's only civil,' answered Bess, affecting
        id=40

words:
1. 'affecting': 11 documents, 12 hits
```

รูปที่ 4.8 แสดงการค้นหาด้วย Sphinx

บทที่ 5

สรุปผลการทดลอง

5.1 ผลการทดลอง

ผลการทำงานเป็นค่าเฉลี่ยจากการสร้างดัชนีจำนวน 5 ครั้ง แล้ว ค่า Precision และ Recall ที่ได้ เป็นค่าเฉลี่ยในการค้นหาอย่างละ 5 คำดังที่ได้กล่าวไว้ในบทที่ 4

ตารางที่ 5.1 การทำงานของ Sphinx ในข้อมูลเข้าที่เป็น XML จำนวน 100 ไฟล์

ด้านที่วัด	ค่าต่าง ๆ	
เวลาที่ใช้ในการสร้างดัชนี	0.89 วินาที	
เวลาที่ใช้ในการค้นหา	เมื่อมีข้อมูล	เมื่อไม่มีข้อมูล
Boolean Operators	0.014 วินาที	0.009
Precision	1	-
Recall	1	-
Wildcard Searches	0.23 วินาที	0.15
Precision	0.70	-
Recall	0.93	-
Fuzzy Searches	-	-
Precision	-	-
Recall	-	-
Proximity Searches	0.33 วินาที	0.29 วินาที
Precision	0.78	-
Recall	0.89	-
Range Searches	0.23 วินาที	0.19 วินาที
Precision	1	-
Recall	1	-

ตารางที่ 5.1 (ต่อ)

ด้านที่วัด	ค่าต่าง ๆ	
Boosting a Term	-	
Precision	-	-
Recall	-	-
Grouping	0.022 วินาที	0.018 วินาที
Precision	1	-
Recall	1	-
Escaping Special Characters	0.43 วินาที	0.040 วินาที
Precision	1	-
Recall	1	-

ตารางที่ 5.2 การทำงานของ Lucene ในข้อมูลเข้าที่เป็น PDF จำนวน 100 ไฟล์

ด้านที่วัด	ค่าต่าง ๆ	
เวลาที่ใช้ในการสร้างดัชนี	12.189 วินาที	
เวลาที่ใช้ในการค้นหา	เมื่อมีข้อมูล	เมื่อไม่มีข้อมูล
Boolean Operators	0.43 วินาที	0.034
Precision	1	-
Recall	1	-
Wildcard Searches	0.86 วินาที	0.16
Precision	0.76	-
Recall	0.89	-
Fuzzy Searches	2.23 วินาที	1.23
Precision	0.35	-
Recall	0.75	-
Proximity Searches	0.45 วินาที	0.19
Precision	0.73	-
Recall	0.87	-

ตารางที่ 5.2 (ต่อ)

ด้านที่วัด	ค่าต่าง ๆ	
Range Searches	1.21 วินาที	0.43
Precision	0.94	-
Recall	0.85	-
Boosting a Term	0.45 วินาที	0.20
Precision	0.91	-
Recall	0.96	-
Grouping	0.65 วินาที	0.19
Precision	1	-
Recall	1	-
Escaping Special Characters	1.98 วินาที	0.74
Precision	1	-
Recall	1	-

เปรียบเทียบการทำงานของ Lucene และ sphinx โดยอ่านจาก ฐานข้อมูล จำนวน 100 เรคคอร์ด

ตารางที่ 5.3 สรุปการทำงานของ Lucene และ Sphinx

ด้านที่วัด	ค่าต่าง ๆ	
	Lucene	Sphinx
เวลาที่ใช้ในการสร้างดัชนี	ช้ากว่า	เร็วกว่า
เวลาที่ใช้ในการค้นหา	ช้ากว่า	เร็วกว่า
การอ่านข้อมูลจากฐานข้อมูล	ทำได้	ทำได้
การอ่านจากไฟล์	PDF,DOC,TXT,XML	XML

5.2 อุปสรรคในการเปรียบเทียบ

5.2.1 เครื่องมือทั้งสองสร้างมาเพื่อการเรียกใช้ภาษาที่ต่างกัน ทำให้เมื่อต้องนำมาใช้ในภาษาที่เหมือนกันนั้น ต้องหยิบเอาเครื่องมือที่มีการปรับปรุงจากคนอื่นมาใช้ ซึ่งอาจจะทำให้ประสิทธิภาพในการใช้งานต่างกันได้

5.2.2 ศึกษาการใช้เครื่องมือทั้งสองชนิดยังไม่ละเอียด ทำให้ผลการรันอาจจะไม่สมบูรณ์

5.2.3 ข้อมูลที่นำมาใช้ยังน้อยเกินไป ทำให้ผลอาจจะไม่แตกต่างกันมากนัก

5.2.4 ความสามารถของเครื่องมือบางชนิดไม่เหมือนกันทำให้อาจต้องเพิ่มขั้นตอนในการสร้างดัชนีข้อมูล เช่น Sphinx ไม่สามารถอ่านข้อมูลจากไฟล์ต่าง ๆ ได้โดยตรง ต้องใช้ภาษาจาวาติดต่อฐานข้อมูลเพื่ออ่านข้อมูลออกมาก่อน ถึงจะอ่านข้อมูลเพื่อสร้างดัชนีได้

5.3 สรุปผลการค้นหา

5.3.1 ทางด้านการสร้างดัชนี

Lucene ใช้เวลาในการสร้างดัชนีมากกว่า Sphinx เนื่องจากว่า Lucene มีการ ตั้งค่าเพื่อเตรียมใช้ในการ ค้นหาได้มากกว่า Sphinx ทำให้มีขั้นตอนในการสร้างเยอะกว่า ส่งผลให้ใช้เวลาในสร้างเยอะกว่า Sphinx

5.3.2 ทางด้านการค้นหา

จากผลการทดลองสรุปได้ว่า Lucene สามารถค้นหาได้ในหลายรูปแบบมากกว่า Sphinx อาจจะมากจากการพัฒนาที่ยาวนานกว่า หรือสร้างมาจาก ภาษาจาวา ที่มีความทันสมัยกว่า ภาษาซี

5.3.3 การสนับสนุน ฐานข้อมูล

ใน Lucene ฉบับดั่งเดิม ยังไม่สามารถอ่านข้อมูลจากฐานข้อมูลได้ ต้องใช้ภาษาจาวา ติดต่อฐานข้อมูลอ่านออกมาก่อน แต่ใน Sphinx การสนับสนุนฐานข้อมูลทำได้อย่างดี แต่ข้อเสียคือ ไม่สามารถใส่ ข้อมูลลงในฐานข้อมูลได้ทีละมาก ๆ

5.3.4 การสนับสนุน ไฟล์

ใน Lucene สามารถอ่านข้อมูลได้หลากหลายไฟล์มากกว่า ใน Sphinx ที่สนับสนุนได้เพียงรูปแบบของ XML เท่านั้น

5.3.5 การนำไปใช้งาน

Sphinx เหมาะกับการหาข้อมูลในรูปแบบเว็บที่ต้องอ่านข้อมูลจากฐานข้อมูล ส่วน Lucene เหมาะกับการหาเนื้อหาในเอกสารที่เป็น ไฟล์ เพราะสนับสนุนไฟล์ได้หลายชนิดกว่า

5.4 ข้อเสนอแนะ

6.4.1 สามารถนำเอาเครื่องมืออื่น ๆมาเปรียบเทียบได้อีก

6.4.2 สามารถนำเอาภาษาอื่น ๆ มาใช้ในการเปรียบเทียบ

6.4.3 สามารถเปรียบเทียบเครื่องมือบนระบบปฏิบัติการอื่น

6.4.4 นำเอาไปใช้ในการพัฒนาระบบค้นหา

6.4.5 สามารถนำเอาฐานข้อมูลอื่น มาตรวจสอบ

6.4.6 สามารถใช้ภาษาอื่น ๆในการเรียกใช้เครื่องมือเพื่อเปรียบเทียบ

6.4.7 สามารถแสดงผลออกเป็นรายงานการค้นหา

บรรณานุกรม

Abbas Ali. 2011. **Sphinx Search Beginner's Guide**. Packt Publishing Ltd.

Gospodnetic Otis and Hatcher Erik forward by Doug Cutting. 2005. **Lucene in Action**. Manning Publication Co.

Richardson W. Clay , Avondolio Donald, Vitale Joe, Len Peter and Smith Kevin T. rt.al. 2004. **Professional Portal Dev.** Wiley Technology Publishing

The Apache Software Foundation. 2004. **Apache Lucene.** http://lucene.apache.org

ภาคผนวก ก ตัวอย่างไฟล์ ข้อมูล

ไฟล์	หัวข้อ
Sdarticle1	Instantiating abstract argumentation with classical Postulates and properties
Sdarticle2	Efficient solutions to factored MDPs with imprecise transition probabilities
Sdarticle3	Local closed world reasoning with description logics under the ell-founded semantics
Sdarticle4	Hybrid trancability of valued constraint problems
Sdarticle5	Local Search with edge weighting and configuration checking heuristics for minimum vertex cover
Sdarticle6	Inconsistent heuristics in theory and practice
Sdarticle7	Learning qualitative models from numerical data
Sdarticle8	Voting almost maximizes social welfare despite limited communication
Sdarticle9	Towards a model of musical interaction and communication
Sdarticle10	SIP Security and the IMS core
Sdarticle11	Dedekind categories with cutoff operators
Sdarticle12	Ambiguous representation as fuzzy relations between sets
Sdarticle13	L-topological spaces as spaces of points
Sdarticle14	Quantitative domains via fuzzy sets: Part II : Fuzzy Scott topology on fuzzy directed-complete posets
Sdarticle15	Fuzzy algebras as a framework for fuzzy topology
Sdarticle16	Selected papers of the Refinement Workshop Turku(2008)
Sdarticle17	Simulation refinement for concurrency verification
Sdarticle18	Completeness of fair ASM refinement
Sdarticle19	A tactic language for refinement of state-rich concurrent specifications
Sdarticle20	Signal processing
Sdarticle21	Variable selection in linear regression: Several approaches based on normalized maximum likelihood
Sdarticle22	Intracranial subdural hematoma as a cause of postoperative delirium and headache in cervical laminoplasty: A case report and review of the literature

ไฟล์	หัวข้อ
Sdarticle23	A robust audio watermarking scheme based on reduced singular value decomposition and distortion removal
Sdarticle24	An ESPRIT-like algorithm for coherent DOA estimation based on data matrix decomposition in MIMO radar
Sdarticle25	Reconstruction of aperiodic FRI signals and estimation of the rate of innovation based on the state space method
Sdarticle26	Output SNR analysis of integrated active noise control and noise reduction in hearing aids under a single speech source scenario
Sdarticle27	Synthesis of multivariate stationary series with prescribed marginal distributions and covariance using circulant matrix embedding
Sdarticle28	Local object – based super-resolution mosaicking from low –resolution video
Sdarticle29	Source localization for multiple speech source using low complexity non-parametric source separation and clustering
Sdarticle30	Iterative methods for the canonical decomposition of multi-way arrays: Application to blind underdetermined mixture identification
Sdarticle31	Filtering for sampled-data stochastic systems with limited capacity channel
Sdarticle32	Group delay reduction in FIR digital filters
Sdarticle33	Postprocessing and sparse blind source separation of positive and partially overlapped data
Sdarticle34	Exponential H filtering for time –varying delay system: Markovian approach
Sdarticle35	Robust video watermarking based on affine invariant regions in the compressed domain
Sdarticle36	The trinion Fourier transform of color images
Sdarticle37	A video steganalytic algorithm against motion-vector-based steganography
Sdarticle38	Adaptive algorithms for sparse system identification
Sdarticle39	A Baysian Lasso via reversible-jump MCMC
Sdarticle40	Fast optimization for multichannel total variation minimization with non-quadratic fidelity

ไฟล์	หัวข้อ
Sdarticle41	Subspace based blind identification of LTI FIR MIMO systems and equalization of finite memory SIMO Volterra systems
Sdarticle42	Optimum crossing – point estimation of a sampled analog signal with a periodic carrier
Sdarticle43	Spatially constrained ICA algorithm with an application in EEG processing
Sdarticle44	Blind and robust audio watermarking scheme based on SVD-DCT
Sdarticle45	A box constrained gradient projection algorithm for compressed sensing
Sdarticle46	LSA based multi-instance algorithm for image retrieval
Sdarticle47	A novel all-neighbor fuzzy association approach for multitarget tracking in a cluttered environment
Sdarticle48	Weighting for more: Enhancing characteristic-function based ICA with asymptotically optimal weighting
Sdarticle49	On parameter identification of MIMO radar with waveform diversity
Sdarticle50	Direct prediction – and smoothing – based Kalman and particle fliter algorithms
Sdarticle51	Self-tuning distributed measurement fusion Kalman estimator for the multi-channel ARMA signal
Sdarticle52	Receiver-side nonlinearities mitigation using an extended iterative decision – bases technique
Sdarticle53	Stability analysis of adaptive with regression vector nonlinearities
Sdarticle54	Context – adaptive pre-processing scheme for robust speech recognition in fast-varying noise environment
Sdarticle55	Three-dimensional reduced-dimention transformation for MIMO radar space-time adaptive processing
Sdarticle56	Discrete-time chaotic systems synchronization performance under additive noise
Sdarticle57	Performance analysis of a two-stage Rao Detector
Sdarticle58	Reduced biquaternion canonical transform convolution and correlation
Sdarticle59	Higher-order moments for musical genre classification

ไฟล์	หัวข้อ
Sdarticle60	Color image canonical correlation analysis for face feature extraction and recognition
Sdarticle61	Direct positioning of stationary targets using MIMO radar
Sdarticle62	Expression transfer for facial sketch animation
Sdarticle63	On the asymptotic distribution of GLR for impropriety of complex signals
Sdarticle64	A tutorial overview on the properties of the discrete cosine transform for encoded in=mage and video processing
Sdarticle65	A novel wideband DOA estimator on Khatri-Rao subspace approach
Sdarticle66	Image analysis by Gaussian- Hermite moments
Sdarticle67	Robust M- periodogram with dichotomous search
Sdarticle68	A three-layer scheme for m-channel multiple description image coding
Sdarticle69	Asynchronous particle filter for tracking using non-synchronous sensor networks
Sdarticle70	A tracker – aware detector threshold optimization formulation for tracking maneuvering target in clutter
Sdarticle71	Connectivity of projected high dimensional data charts on one-dimensional curves
Sdarticle72	Direct data domain STAP using Representation of Clutter Spectrum
Sdarticle73	Registration – based Compensation using Sparse Representation in Conformal-array STAP
Sdarticle74	Geometric MMSE for one-sided and two – sided vector linear predictors: from the finite-length case to the infinite-length case
Sdarticle75	Joint tracking and discrimination of exoatmospheric active decoys using 9-dimensional parameter-augmented EKF
Sdarticle76	3D human posture segmentation by spectral clustering with surface normal constraint
Sdarticle77	Fourier spectral factor model for prediction of multidimensional signals
Sdarticle78	Activelets: Wavelets for sparse representation of hemodynamic responses
Sdarticle79	Computing the polyadic decomposition of nonnegative third order tensors

ไฟล์	หัวข้อ
Sdarticle80	Bandlet image estimation with model selection
Sdarticle81	Fast orthogonal sparse approximation algorithms over local dictionaries
Sdarticle82	Resonance-based signal decomposition: A new sparsity-enabled signal analysis method
Sdarticle83	A review of medical error taxonomies : A human factors perspective
Sdarticle84	Occupational risk assessment in construction industry-Overview and reflection
Sdarticle85	The relationship between the implementation of a Safety Management System and the Attitudes of employees towards unsafe acts in aviation
Sdarticle86	The development of probabilistic models to estimate accident risk (due to runway overrun and landing undershoot) applicable to the design and construction of runway safety areas
Sdarticle87	Prediction of the confidence interval for stability of chain pillars in coal mines
Sdarticle88	Analysis of "high-dollar" value safety and health citations and orders for the US coal mines
Sdarticle89	Reliability estimation of auxiliary ventilation systems in long tunnels during construction
Sdarticle90	A mathematical model on adjacent smoke filling involved sprinkler cooling to a smoke layer
Sdarticle91	Designing a developed model for assessing the disaster induced vulnerability value in educational centers
Sdarticle92	Methodology for consequence analysis of LNG releases at deepwater port facilities
Sdarticle93	A systemic methodology for risk management in healthcare sector
Sdarticle94	Visibility – related fatalities related to construction equipment
Sdarticle95	Assessing gait changes in firefighters due to fatigue and protective clothing

ไฟล์	หัวข้อ
Sdarticle96	An efficient framework of emergency response to facilitate disaster recovery for fire-damaged medical equipment – Case study at a large medical center after a fire
Sdarticle97	Identifying common problems in the acquisition and deployment of large=scale safety – critical software projects in the US and UK healthcare systems
Sdarticle98	Indices of ergonomic-psycholsociological workplace quality in the greenhouses of Almeria (Spain): Crops of cucumbers. Peppers. Aubergines and melons
Sdarticle99	Driver Behaviour and Accident Research Methodology Unresolved Problems
Sdarticle100	Injury of the spinal cord due to rupture of an intervertebral disk during muscular effort

ภาคผนวก ข รูปแบบไฟล์ที่ได้จากการ สร้าง index ของ Lucene

ชื่อ	ไฟล์	อธิบาย
Segments	Segments.gen, segments_N	เก็บข้อมูลเกี่ยวกับ segment
Lock File	Write.lock	ล๊อกไฟล์ที่มีการเขียนไฟล์เดียวกันจากหลาย ๆ ที่
Compound File	.cfs	ประกอบด้วยไฟล์ดัชนีต่าง ๆของระบบที่บอกความถี่ในการจัดการไฟล์
Fields	.fnm	เก็บข้อมูลเกี่ยวกับ ฟิลด์
Field Index	.fdx	เก็บตัวชี้เพื่อชี้ไปที่ข้อมูลที่เก็บไว้
Field Data	.fdt	เก็บฟิลด์สำหรับเอกสาร
Term Infos	.tis	ส่วนของความหมายคำ เก็บข้อมูลคำ
Term Info Index	.tii	ดัชนีที่บอกถึงไฟล์ของคำศัพท์
Frequencies	.frq	เก็บรายการเอกสารแต่ลำคำพร้อมความถี่
Positions	.prx	เก็บข้อมูลตำแหน่งคำอยู่ที่ไหนในดัชนี
Norms	.nrm	เก็บขนาดและปัจจัยเสริมอื่นของเอกสารและฟิลด์
Term Vector Index	.tvx	เก็บตำแหน่งในไฟล์ข้อมูลเอกสาร
Term Vector Document	.tvd	เก็บข้อมูลเกี่ยวกับเอกสารแต่ละอันว่าที่ทิศทางของคำอย่างไร
Term Vector Fields	.tvf	ระดับของฟิลด์ที่เกี่ยวกับทิศทางของคำ
Deleted Documents	.del	เก็บข้อมูลไฟล์ที่ถูกลบ

ประวัติผู้เขียน

ชื่อผู้เขียน	นางสาว สุดาวรรณ์ ธัญญสกุลกิจ
วันเดือนปีเกิด	21 ตุลาคม 2524
สถานที่เกิด	จังหวัดนครปฐม
วุฒิการศึกษาระดับปริญญาตรี	วิทยาศาสตร์บัณฑิต
สถานที่สำเร็จการศึกษา	ภาควิชา วิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์
ปีการศึกษาที่สำเร็จการศึกษา	2547
ความชำนาญเฉพาะด้าน	1. เขียน PL/SQL procedure 2. ระบบ billing