การพัฒนาระบบค้นคืนข้อมูลโดยลูซีน
(Developing IR Systems via Lucene)

ชูชาติ หฤไชยะศักดิ์
งานวิจัยและพัฒนาโครงสร้างพื้นฐานสารสนเทศอัจฉริยะ (RDI-5)
ฝ่ายวิจัยและพัฒนาเทคโนโลยีสารสนเทศ (RDI)
ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (เนคเทค)
112 อุทยานวิทยาศาสตร์ประเทศไทย ถ.พหลโยธิน อ.คลองหลวง จ.ปทุมธานี 12120
Email: rdi5@nectec.or.th

คำนำ:

เอกสารฉบับนี้ได้จัดทำขึ้นมาเพื่อให้ความรู้และอธิบายการใช้งานลูซีนซึ่งเป็นต้นรหัสซอฟต์แวร์สำหรับใช้พัฒนาระบบและแอพพลิเคชั่นที่เกี่ยวกับการค้นคืนข้อมูล (Information Retrieval) เช่น ระบบสืบค้นข้อมูลบนอินเตอร์เน็ต (Internet Search Engines) และระบบค้นคืนข้อมูลภายในองค์กร (Intranet Information Retrieval Systems) เป็นต้น ลูซีน (Lucene – lucene.apache.org) ได้รับการพัฒนาและนำไปใช้งานสำหรับการค้นคืนข้อมูลอย่างกว้างขวางและต่อเนื่อง ผู้ที่ริเริ่มพัฒนาลูซีนขึ้นมา คือ ดัก คัตติ้ง (Doug Cutting) นักวิจัยและพัฒนาโปรแกรมซึ่งมีความรู้และความเชี่ยวชาญทางด้านเทคนิคการค้นคืนข้อมูลเป็นอย่างดี ลูซีนที่ได้รับการพัฒนาในยุคแรกนั้นอยู่ในภาษาจาวา (Java) ต่อมาในปี ค.ศ. 2001 ลูซีนได้รับความสนใจจากผู้ใช้เป็นจำนวนมากขึ้น ทำให้ลูซีนถูกนำไปเป็นส่วนหนึ่งของโครงการจาการ์ต้า (Jakarta Project) ซึ่งอยู่ภายใต้โครงการซอฟต์แวร์อาพาเช่ (Apache Software Foundation) เนื่องจากเป้าหมายหลักของโครงการคือสนับสนุนการพัฒนาซอฟต์แวร์ในรูปแบบต้นรหัสเปิด (Open Source Software) ดังนั้น ลูซีนจึงได้รับการพัฒนาอย่างต่อเนื่อง ทั้งการแก้ไขข้อผิดพลาดของการทำงาน รวมทั้งมีการเพิ่มความสามารถต่างๆให้มากยิ่งขึ้น

ในปัจจุบันมีการนำลูซีนไปใช้ประยุกต์ใช้งานในระบบและแอพพลิเคชั่นต่างๆมากมาย ทั้งยังมีการพัฒนาลูซีนในโปรแกรมภาษาต่างๆมากมาย เช่น C, C++, C#, Perl และ Python นอกจากความมีประสิทธิภาพสูงในการสร้างดัชนีและการค้นคืนข้อมูลแล้ว ข้อดีที่สำคัญของลูซีนคือการออกแบบโครงสร้างของต้นรหัสอย่างมีระเบียบ ทำให้นักพัฒนาโปรแกรมสามารถเพิ่มรหัสในส่วนที่ลูซีนยังขาดอยู่ได้ไม่ยากนัก ส่วนหนึ่งที่สำคัญคือการรองรับการค้นคืนข้อความและเอกสารในภาษาต่างๆ เนื่องจากลูซีนได้รับการพัฒนาเพื่อรองรับการใช้งานกับภาษาอังกฤษเป็นหลัก ดังนั้นเมื่อนักพัฒนาโปรแกรมนำเอาลูซีนไปประยุกต์ใช้งานสำหรับภาษาอื่นๆ ก็จำเป็นที่ต้องเพิ่มรหัสในส่วนของการวิเคราะห์ภาษา (Analyzer Package) ด้วย ตัวอย่างของส่วนที่ทำการวิเคราะห์ภาษาอื่นๆที่ทำไปแล้วคือ ภาษาเยอรมันและภาษาจีน แต่ยังไม่มีการพัฒนาในส่วนของภาษาไทย

ทีมนักวิจัยและพัฒนาโปรแกรมสรรสารซึ่งเป็นหน่วยงานวิจัยโครงสร้างพื้นฐานสารสนเทศอัจฉริยะ ของฝ่ายวิจัยและพัฒนาเทคโนโลยีสารสนเทศ (RD-I) สังกัดศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ได้มีการทำวิจัยและพัฒนาเทคนิคต่างๆที่เกี่ยวข้องเทคโนโลยีค้นคืนข้อมูลมานานหลายปี ผลงานที่ผ่านมาคือระบบสืบค้นข้อมูลบนอินเตอร์เน็ตในชื่อสรรสาร (sansarn.com) ที่ผ่านมาทางทีมสรรสารได้ช่วยพัฒนาระบบสืบค้นข้อมูลให้กับหน่วยงานทั้งของรัฐและเอกชนต่างๆมากมาย แม้ว่าทางทีมสรรสารได้ช่วยพัฒนาระบบสืบค้นข้อมูลให้กับหน่วยงานต่างๆ แต่ความต้องการใช้งานของระบบสืบค้นข้อมูลก็ยังมีอยู่มากและทางทีมงานซึ่งมีนักวิจัยและพัฒนาเพียงไม่กี่คนก็ไม่สามารถรับพัฒนาระบบให้แก่ผู้ที่ต้องการใช้ระบบเป็นจำนวนมากได้ ดังนั้นทางทีมจึงมีแนวทางที่จะพัฒนาระบบค้นคืนข้อมูลในลักษณะต้นแบบรหัสเปิด (Open Source Search Engines) โดยใช้ลูซีนเป็นฐาน โดยทางทีมสรรสารซึ่งมีความรู้และความเชี่ยวชาญในการวิเคราะห์และประมวลผลภาษาไทยได้ทำการพัฒนาต่อยอดในส่วนของการวิเคราะห์ภาษาไทย (ThaiAnalyzer) สำหรับการใช้งานลูซีนโดยเฉพาะ

ทางทีมสรรสารหวังเป็นอย่างยิ่งว่าผู้พัฒนาโปรแกรมและเว็บมาสเตอร์ รวมทั้งนักศึกษาทั่วไปสามารถที่จะนำเอาลูซีนที่มีส่วนวิเคราะห์ภาษาไทย (ThaiAnalyzer) ไปใช้พัฒนาต่อยอดในระบบค้นคืนข้อมูลต่างๆ โดยที่ไม่ต้องเสียค่าธรรมเนียมการใช้โปรแกรม (License Fee) ซึ่งจะช่วยให้การใช้งานเทคโนโลยีสืบค้นข้อมูลในประเทศไทยเป็นไปอย่างแพร่หลายมากขึ้น และช่วยลดการนำเข้าของซอฟต์แวร์ต่างประเทศได้อีกด้วย

<u>หัวข้อในบทความนี้:</u>

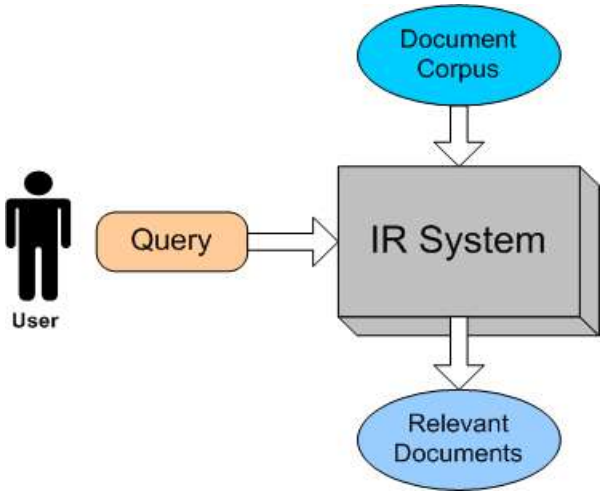
1. **เทคนิคการค้นคืนข้อมูลทั่วไป (Information Retrieval Techniques)**: ในส่วนนี้จะเป็นความรู้เบื้องต้นเกี่ยวกับเทคนิค และส่วนประกอบต่างๆ รวมทั้งขั้นตอนในการพัฒนาระบบค้นคืนข้อมูลทั่วๆไป ผู้ที่มีพื้นฐานเกี่ยวกับระบบค้นคืนข้อมูลสามารถข้ามเนื้อหาในส่วนนี้ได้
2. **การประยุกต์ใช้ลูซีนสำหรับพัฒนาระบบค้นคืนข้อมูล (Lucene for IR Applications)**: ในส่วนนี้จะเป็นการแนะนำการนำเอาลูซีนไปประยุกต์ใช้พัฒนาระบบค้นคืนข้อมูลในเบื้องต้น
3. **โครงสร้างทางสถาปัตยกรรมของลูซีน (Lucene API)**: ในส่วนนี้จะอธิบายรายละเอียดของโครงสร้างและส่วนประกอบทางสถาปัตยกรรมของลูซีน โดยจะมีตัวอย่างการเรียกใช้งานของโปรแกรมในแต่ละส่วน และจะกล่าวถึง ThaiAnalyzer ซึ่งมีหน้าที่ในการวิเคราะห์ข้อความภาษาไทย โดยมีผลของการเปรียบเทียบการใช้ Analyzer แบบต่างๆ ด้วย
4. **สาธิตการใช้งานลูซีน (Lucene Demostration)**: ในส่วนนี้จะเป็นการสาธิตการเรียกใช้งานลูซีน โดยจะมีรหัสโปรแกรมที่ใช้สร้างดัชนีจากเอกสารที่อยู่ในระบบไฟล์ของผู้ใช้และมีส่วนของการค้นคืนเอกสารโดยใช้คิวรี่ นักพัฒนาระบบสามารถเรียนรู้จากโปรแกรมในส่วนนี้แล้วนำเอารหัสโปรแกรมไปพัฒนาต่อยอดได้ต่อไป
5. **แนะนำโปรแกรมสรรสารลุ๊ค (Introduction to Sansarn Look!)**: นอกจากในส่วนของ ThaiAnalyzer ที่ทางทีมสรรสารได้พัฒนาขึ้นมาแล้ว ในขณะนี้ทางทีมก็กำลังพัฒนารหัสโปรแกรมในลักษณะเป็นโมดูลที่เรียกใช้ลูซีน โมดูลเหล่านี้เป็นส่วนที่ทำหน้าที่หลักๆในระบบค้นคืนข้อมูลทั่วไปเช่น ส่วนที่ทำหน้าที่สร้างดัชนี ส่วนที่เกี่ยวข้องกับการค้นคืน เป็นต้น นักพัฒนาระบบสามารถสร้างระบบโดยเรียกใช้โปรแกรมต่างๆเหล่านี้โดยไม่ต้องเสียเวลาเขียนโปรแกรมขึ้นมาเอง

<u>1. เทคนิคการค้นคืนข้อมูลทั่วไป</u> (Information Retrieval Techniques)

ความจำเป็นในการค้นคืนข้อมูลในรูปแบบดิจิตอลไม่ได้เพิ่งจะเริ่มต้นในยุคเทคโนโลยีสารสนเทศ (Information Technology) แต่ย้อนหลังไปตั้งแต่ช่วงตั้งแต่ราวทศวรรษ 1960 แต่ในช่วงแรกการค้นคืนข้อมูลยังจำกัดอยู่เพียงเอกสารจำนวนไม่มากเท่าไร และส่วนมากยังอยู่ในโดเมนที่จำกัด เช่น เอกสารทางด้านกฎหมายและธุรกิจ เป็นต้น ในหลายทศวรรษต่อมาเอกสารเริ่มมีจำนวนมากขึ้นตามลำดับ เนื่องมาจากการแพร่หลายของการใช้งานคอมพิวเตอร์ในการพิมพ์เอกสารแทนเครื่องพิมพ์ดีด อย่างไรก็ตามการเกิดขึ้นของอินเตอร์เน็ตและการใช้งานกันอย่างแพร่หลายของเครือข่ายข้อมูล WWW (World Wide Web) ในช่วงทศวรรษที่ 1990 เป็นปัจจัยหลักที่ทำให้ข้อมูลมีปริมาณเพิ่มขึ้นอย่างมากมายและต่อเนื่อง ความจำเป็นที่ตามมาอย่างหลีกเลี่ยงไม่ได้คือความจำเป็นในการค้นหาเอกสารหรือข้อมูลที่ต้องการในเวลาอันรวดเร็ว วิวัฒนาการต่างๆเหล่านี้ทำให้มีการศึกษาและวิจัยเทคนิคการค้นคืนข้อมูลที่มีประสิทธิภาพมากพอในการรองรับกับปริมาณข้อมูลจำนวนมาก

โดยทั่วไประบบค้นคืนข้อมูลจะใช้เทคนิคการค้นคืนโดยใช้คำสำคัญ (Keyword-Based Retrieval) หลักการทำงานคือผู้ใช้ต้องระบุคำต่างๆที่ต้องการค้นหา จากนั้นระบบจะค้นคืนผลลัพธ์เป็นรายการเอกสารที่มีคำเหล่านั้นปรากฏอยู่ รูปที่ 1 แสดงภาพรวมของระบบค้นคืนข้อมูลโดยทั่วไป การทำงานเริ่มจากระบบทำการสร้างดัชนีจากชุดเอกสารที่ต้องการค้นคืน ดัชนี (Indexes) คือโครงสร้างข้อมูล (Data Structure) ที่จัดเตรียมไว้ล่วงหน้าซึ่งทำให้การค้นคืนเป็นไปอย่างรวดเร็ว โดยทั่วไปดัชนีจะเป็นกลุ่มของคำต่างๆพร้อมทั้งหมายเลขของเอกสารที่คำเหล่านี้ปรากฏอยู่ ผู้ใช้สามารถค้นหาเอกสารที่

ต้องการโดยการระบุชุดของคำ (Query) ที่ต้องการ จากนั้นระบบจะทำการตรวจสอบหาเอกสารที่มีคำต่างๆที่ผู้ใช้ระบุปรากฏอยู่จากดัชนี และเสนอเป็นรายการของเอกสารให้ผู้ใช้ต่อไป



รูปที่ 1: การทำงานของระบบค้นคืนข้อมูลทั่วๆไป

ระบบค้นคืนข้อมูลทั่วไปมีส่วนประกอบหลัก 6 ส่วนที่สำคัญ ได้แก่ (รูปที่ 2)

1) หน่วยประมวลข้อความ (Text Processing Module): มีหน้าที่ในการวิเคราะห์และประมวลข้อความสำหรับการสร้างดัชนีคำ (Index Terms) ตัวอย่างการทำงานในส่วนนี้ คือ

- การตรวจการเรียงตัวของอักขระและการตัดคำ (Parsing and Tokenizing): ส่วนนี้จำเป็นอย่างยิ่งสำหรับภาษาที่เขียนเรียงติดกัน (Non-Segmented Languages) อย่างเช่น ภาษาจีน ญี่ปุ่น และไทย เป็นต้น
- การแปลงคำให้อยู่ในรูปของรากศัพท์ (Stemming and Plural Removal): ส่วนนี้ใช้มากสำหรับภาษาแบบละติน (Latin-Based Languages) เช่น ภาษาอังกฤษ อิตาลี และ สเปน เป็นต้น ตัวอย่างเช่น คำว่า "computed" "computer" "computing" จะถูกแปลงเป็น "compute" ทั้งหมด หรือ คำว่า "properties" แปลงเป็น "property" เป็นต้น
- การกรองเอาคำที่ไม่มีความหมายสำคัญและเกิดขึ้นบ่อยๆออก (Stopword Removal): เช่น คำว่า "the" "on" "from" ในภาษาอังกฤษ หรือ คำว่า "การ" และ "ที่" ภาษาไทย เป็นต้น

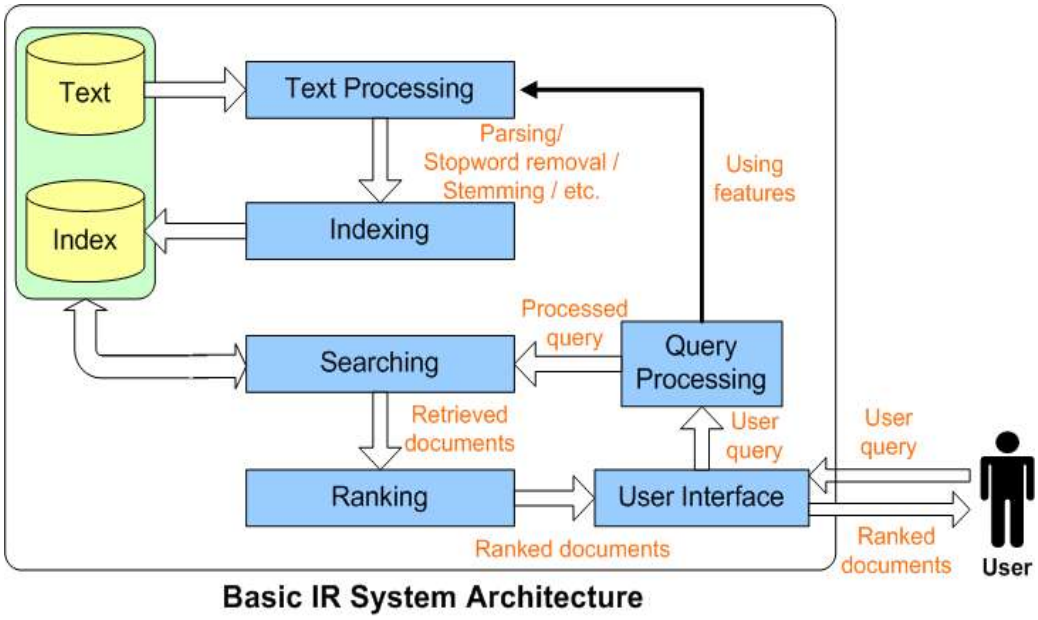
2) หน่วยสร้างดัชนี (Indexing Module): มีหน้าที่ในการสร้างชุดดัชนีของคำที่ได้จากการวิเคราะห์และประมวลข้อความ โดยทั่วไปดัชนีจะเป็นกลุ่มของคำต่างๆพร้อมทั้งหมายเลขของเอกสารที่คำเหล่านี้ปรากฏอยู่

3) หน่วยค้นคืนเอกสาร (Searching Module): มีหน้าที่ในการค้นคืนรายการเอกสารโดยการตรวจสอบคำที่ผู้ใช้ระบุกับชุดดัชนีคำที่ระบบสร้างเก็บไว้ล่วงหน้า

4) หน่วยจัดลำดับผลลัพธ์จากการค้นคืน (Ranking Module): มีหน้าที่ในการจัดเรียงลำดับของเอกสารที่ได้จากการค้นคืน ในทางทฤษฎีเอกสารที่เกี่ยวข้องกับที่ผู้ใช้ต้องการมากที่สุด (high relevance) ในขณะนั้นควรจะถูกเสนอให้ผู้ใช้เป็นอันดับแรกๆ แต่ในทางปฏิบัติเป็นการยากที่จะวัดความต้องการของผู้ใช้ที่แท้จริง เทคนิคการจัดเรียงลำดับของผลลัพธ์เอกสารโดยทั่วไปส่วนใหญ่จะดูจากความถี่ของคำในเอกสาร เวลาที่แก้ไขเอกสารครั้งล่าสุด หรืออาจจะมีการวิเคราะห์ความนิยมในการถูกอ้างอิงจากเอกสารอื่นๆ เช่นในกรณีของเอกสารบนเว็บ (Web Pages) เป็นต้น

5) หน่วยประมวลคิวรี่จากผู้ใช้ (Query Processing Module): คิวรี่ (query) เป็นชุดของคำพร้อมทั้งสัญลักษณ์ในการเชื่อมคำเช่นเครื่องหมายตรรกะ "AND" "OR" และ "NOT" เป็นต้น ผู้ใช้ส่งคิวรี่ให้แก่ระบบจากนั้นระบบต้องทำการประมวลคิวรี่เพื่อให้รู้ความต้องการค้นคืนของผู้ใช้

6) หน่วยเชื่อมต่อระหว่างผู้ใช้และระบบ (User Interface Module): มีหน้าที่ในการรับคิวรี่จากผู้ใช้และส่งผ่านให้ระบบประมวลผล และยังมีหน้าที่ในการแสดงผลลัพธ์จากการค้นคืนให้กับผู้ใช้

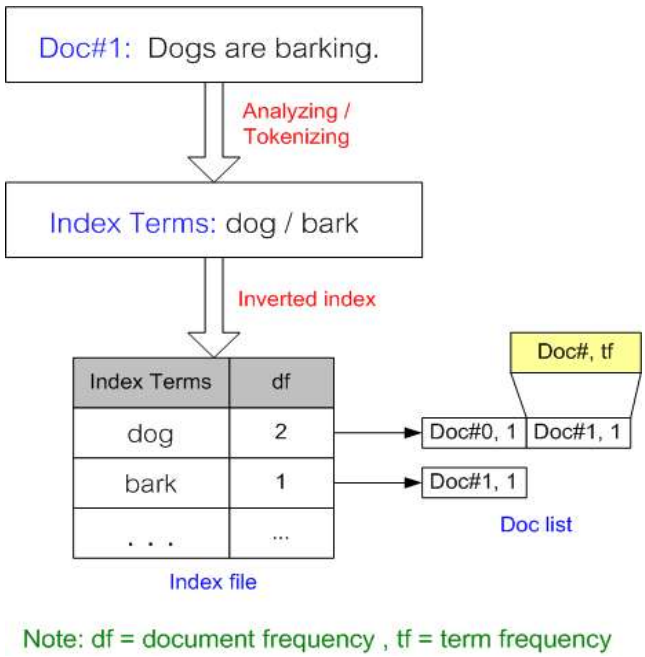


รูปที่ 2: โครงสร้างพื้นฐานทางสถาปัตยกรรมของระบบค้นคืนข้อมูล

การทำงานของระบบค้นคืนข้อมูลทั่วไปสามารถแบ่งออกเป็น 2 ขั้นตอนที่สำคัญ คือ

1) **ขั้นตอนการสร้างดัชนี (Indexing Phase):** ในขั้นตอนนี้ระบบจะทำการสร้างดัชนีจากคลังเอกสาร (Text or Document Corpus) โดยที่เอกสารที่จะนำมาสร้างดัชนีต้องนำมาผ่านในส่วนสำหรับประมวลข้อความ (Text Processing Module) ก่อนเพื่อสกัดเอาคำที่สำคัญไปสร้างดัชนีคำ เมื่อมีการเปลี่ยนแปลงและแก้ไขเอกสาร หรือมีการเพิ่มเอกสาร ระบบจะต้องทำการสร้างดัชนีใหม่เพื่อให้สอดคล้องกับเอกสารชุดใหม่ ทั้งนี้เทคนิคการสร้างดัชนีที่มีประสิทธิภาพจะต้องมีความสามารถในการเพิ่มดัชนีในส่วนที่เปลี่ยนไปโดยไม่กระทบกับส่วนที่มีอยู่แล้ว เทคนิคนี้มีชื่อว่า Incremental Indexing ซึ่งทำให้ไม่ต้องเสียเวลาในการแก้ไขข้อมูลดัชนีใหม่ทั้งหมดในกรณีที่มีการเปลี่ยนแปลงของเอกสารเพียงเล็กน้อย

   เทคนิคการสร้างดัชนีของลูซีนนั้นเป็นแบบ Inverted File Index วิธีการสร้างดัชนีแบบนี้เริ่มจากการนำเอกสารมาวิเคราะห์และแบ่งเป็นรายการของคำ (Index Terms) ซึ่งอาจจะมีการกรองเอาคำที่ไม่มีความหมายสำคัญออก (Stopword removal) หรือแปลงคำให้เป็นรากศัพท์ (Stemming) จากนั้นจึงนำมาเก็บเป็นไฟล์ดัชนีซึ่งมีคำต่างๆพร้อมทั้งจำนวนเอกสารที่คำนั้นๆปรากฏอยู่ และแต่ละคำก็จะมีข้อมูลซึ่งระบุรายการหมายเลขของเอกสารพร้อมทั้งจำนวนคำที่ปรากฏอยู่ในเอกสารนั้น (รูปที่ 3)



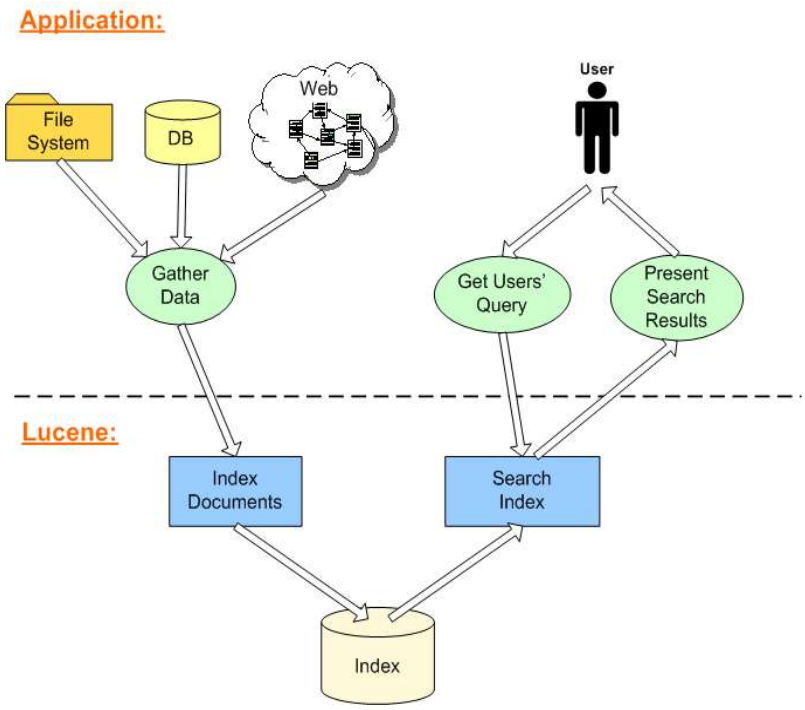
<u>รูปที่ 3: การสร้างดัชนีแบบ Inverted File Index</u>

2) **ขั้นตอนการค้นคืนเอกสาร (Retrieval Phase):** ขั้นตอนนี้เริ่มจากที่ผู้ใช้พิมพ์และส่งคิวรี่ให้กับระบบ หน่วยเชื่อมต่อระหว่างผู้ใช้และระบบจะรับคิวรี่แล้วส่งไปยังหน่วยประมวลคิวรี่จากผู้ใช้ จากนั้นหน่วยค้นคืนจะทำการค้นคืนรายการเอกสารโดยตรวจสอบกับข้อมูลดัชนีคำ เมื่อได้รายการของเอกสารแล้วหน่วยจัดลำดับผลลัพธ์จากการค้นคืนจะจัดเรียงลำดับของเอกสารที่ได้จากการค้นคืนตามความต้องการของผู้ใช้ จากนั้นรายการของเอกสารที่เรียงลำดับแล้วจะถูกส่งไปยังหน่วยเชื่อมต่อระหว่างผู้ใช้และระบบเพื่อแสดงผลแก่ผู้ใช้ต่อไป

<u>2. การประยุกต์ใช้ลูซีนสำหรับพัฒนาระบบค้นคืนข้อมูล</u> (Lucene for IR Applications)

นักพัฒนาโปรแกรมสามารถนำเอาลูซีนไปประยุกต์ใช้พัฒนาระบบค้นคืนข้อมูลหรือแอพพลิเคชั่นที่เกี่ยวข้องกับการค้นคืนข้อมูลได้โดยไม่ยาก รูปที่ 4 แสดงหน้าที่ที่ลูซีนรับผิดชอบเป็นหลัก ซึ่งได้แก่การสร้างฐานข้อมูลดัชนีและการค้นคืนเอกสารจากฐานข้อมูลดัชนีได้อย่างมีประสิทธิภาพมากที่สุด ซึ่งได้แก่เวลาที่ใช้ในการสร้างดัชนีและค้นคืนเอกสาร (Indexing and Searching Time) ปริมาณเนื้อที่ที่ใช้สำหรับจัดเก็บดัชนี (Required Index Storage) ความถูกต้องและครอบคลุมในการค้นคืน (Precision and Recall) โดยที่ผู้ใช้ไม่จำเป็นต้องรู้รายละเอียดในเชิงเทคนิค เพียงแต่รู้โครงสร้างทางสถาปัตยกรรม (API) เพื่อสามารถเรียกใช้ได้อย่างสะดวกก็พอ

ดังนั้นถ้าหากมองว่าลูซีนเป็นส่วนที่อยู่ในชั้นล่างซึ่งเป็นฐานแล้ว นักพัฒนาโปรแกรมมีหน้าที่ในการสร้างระบบหรือแอพพลิเคชั่นซึ่งอยู่ในชั้นบนของลูซีนนั่นเอง ส่วนที่นักพัฒนาต้องรับผิดชอบทำขึ้นมาเองได้แก่ ส่วนที่มีหน้าที่ดึงและจักเก็บเอกสารและข้อมูลต่างๆ (Document Collection) ซึ่งเอกสารและข้อมูลนั้นอาจจะอยู่ในหลากหลายรูปแบบและยังสามารถอยู่ในเครื่องคอมพิวเตอร์ที่ใช้งานอยู่ (Local) หรืออาจจะอยู่ในเครื่องที่เชื่อมต่อในเครือข่าย (Remote) ก็ได้ เช่น ระบบไฟล์ (File System) ฐานข้อมูล (Database) หรือ เอกสารบนเว็บ (Web page) เป็นต้น ดังนั้นการออกแบบโปรแกรมในส่วนนี้ก็ขึ้นอยู่กับลักษณะของข้อมูลที่ต้องการจะค้นหานั่นเอง โปรแกรมในอีกส่วนหนึ่งที่นักพัฒนาต้องทำขึ้นมาเองคือส่วนเชื่อมต่อระหว่างระบบกับผู้ใช้ (User Interface) ซึ่งมีหน้าที่ในการรับคิวรี่จากผู้ใช้และส่งผ่านให้ระบบประมวลผล และยังมีหน้าที่ในการแสดงผลลัพธ์จากการค้นคืนให้กับผู้ใช้ การออกแบบในส่วนนี้ก็ขึ้นอยู่กับลักษณะการใช้งานของระบบ แต่โดยทั่วๆไปปัจจัยหลักในการออกแบบคือการคำนึงถึงผู้ใช้เป็นหลัก ระบบที่ดีควรจะให้ผู้ใช้สามารถค้นคืนได้อย่างง่ายและสะดวกโดยไม่จำเป็นต้องเสียเวลาในการเรียนรู้ในการใช้งานมากนัก



รูปที่ 4: การประยุกต์ใช้ลูซีนสำหรับระบบค้นคืนข้อมูล

3. โครงสร้างทางสถาปัตยกรรมของลูซีน (Lucene API)

ส่วนนี้ประกอบไปด้วยรหัสโปรแกรมพื้นฐานที่เกี่ยวข้องกับการสร้างดัชนีของคำและการค้นคืนเอกสารโดยอาศัยดัชนีที่สร้างไว้ ลูซีนเป็นต้นแบบรหัสโปรแกรมแบบเปิด (Open Source Software) ที่มี API (Application Programming Interface) พร้อมให้นำไปประยุกต์ใช้ในแอพพลิเคชั่นสำหรับค้นคืนข้อมูลและเอกสารในรูปแบบ full-text และยังมีคุณสมบัติต่างๆในการค้นคืนมากมาย รวมทั้งมีประสิทธิภาพในการสร้างดัชนีและค้นคืนได้อย่างรวดเร็ว (high performance) และที่สำคัญคือสามารถรองรับปริมาณเอกสารจำนวนมากได้ดี (good scalability) โครงสร้างทางสถาปัตยกรรมของลูซีนประกอบไปด้วย 4 ส่วนที่สำคัญ คือ (รูปที่ 5)

รูปที่ 5: โครงสร้างทางสถาปัตยกรรมของลูซีน (Lucene API)

- **Document:** มีหน้าที่จัดการโครงสร้างของเอกสารโดยแบ่งเป็น field ต่างๆตามการออกแบบเอกสาร เช่น ชื่อผู้แต่ง วันที่ และ เนื้อหา เป็นต้น
- **Index:** มีหน้าที่สร้างดัชนีของคำจากเอกสาร โดยที่ตัวโปรแกรมนี้มีการใช้เทคนิคและ algorithm ที่ใช้จัดการระหว่างหน่วยความจำหลัก (Main Memory) และหน่วยความจำถาวรอย่างฮาร์ดดิสค์ (hard disk drives) ได้อย่างมีประสิทธิภาพสูง ทำให้กระบวนการสร้างดัชนีเป็นไปอย่างรวดเร็ว
- **Search:** มีหน้าที่ในการค้นคืนข้อมูลโดยอาศัยดัชนีที่สร้างไว้ โดยที่ตัวโปรแกรมนี้มีคุณสมบัติในการรองรับการค้นหา

เอกสารได้หลากหลายรูปแบบ เช่น การค้นคืนจากคำ (Term Query) การค้นคืนแบบตรรกะ (Boolean Query) และการค้นคืนโดยระบุขอบเขต (Range Query) เป็นต้น

- **Analysis:** มีหน้าที่ในการวิเคราะห์ข้อความและสกัดคำเพื่อนำไปสร้างดัชนี โดยที่ตัวโปรแกรมนี้สามารถวิเคราะห์ได้หลายแนวทาง เช่น สามารถตัดคำที่ไม่สำคัญออกได้ (Stopword Removal) เป็นต้น อย่างไรก็ตาม Analysis ที่ให้มากับลูซีนนั้นเน้นการใช้งานกับข้อความและเอกสารในภาษาอังกฤษเป็นหลัก
- **ThaiAnalyzer:** เป็นส่วนที่ทีมสรรสารทำการวิจัยและพัฒนาขึ้นมาเพื่อรองรับการค้นคืนข้อความและเอกสารที่เป็นภาษาไทย ThaiAnalyzer มีหน่วยโปรแกรมที่ทำหน้าที่วิเคราะห์การเรียงตัวของอักขระในภาษาไทย (Parser) และทำการสกัดคำเพื่อนำไปสร้างดัชนี

2.1 Document API

Document เป็น class ที่ประกอบไปด้วย field ต่างๆซึ่งผู้ใช้ระบุสำหรับชุดเอกสารที่นำมาสร้างดัชนี ตัวอย่าง field เช่น ชื่อผู้แต่ง หัวข้อเรื่อง และเนื้อหา เป็นต้น ลูซีนนั้นมีหน้าที่ในการสร้างดัชนีจากข้อความ (texts) เท่านั้น ถ้าหากเป็นเอกสารที่อยู่ในรูปแบบเฉพาะ เช่น .pdf .doc ผู้ใช้จำเป็นต้องใช้ parser หรือโปรแกรมสกัดข้อความที่เหมาะสม แล้วจึงส่งไปจัดให้อยู่ใน Document โดยผ่านในรูปแบบของ java.lang.String หรือ java.io.Reader

Document มี class ที่เกี่ยวข้องคือ Field ซึ่งใช้ในการจัดรูปแบบของเอกสาร ผู้ใช้สามารถค้นหาเอกสารตาม field ได้ ลูซีนมี field อยู่ 4 ชนิดซึ่งใช้สำหรับระบุในการสร้างดัชนี (ตารางที่ 1 สรุปลักษณะ Field แต่ละชนิดพร้อมทั้งตัวอย่างการใช้งาน)

- **Keyword**: จะไม่ผ่านการวิเคราะห์และตัดคำแต่จะถูกนำไปสร้างเป็นดัชนีและจัดเก็บในฐานข้อมูลในรูปแบบเดิมทุกประการ เหมาะสำหรับ field ที่ผู้ใช้ต้องการคงรูปแบบของคำไว้โดยไม่ให้มีการเปลี่ยนแปลง เช่น URLs ที่อยู่ของไฟล์และไดเร็กทอรี่ วันที่ หมายเลขโทรศัพท์ รหัสพนักงาน เป็นต้น
- **UnIndexed**: จะไม่ผ่านทั้งการวิเคราะห์และการสร้างเป็นดัชนี แต่จะถูกจัดเก็บในฐานข้อมูลในรูปแบบเดิมทุกประการ เหมาะสำหรับ field ที่ผู้ใช้ต้องการให้ระบบแสดงพร้อมกับผลลัพธ์ที่ได้จากการค้นคืน แต่ผู้ใช้ไม่สามารถค้นข้อมูลใน field นี้ได้ และไม่เหมาะสำหรับใช้กับข้อมูลขนาดใหญ่
- **UnStored**: มีลักษณะตรงข้ามกับ UnIndexed คือจะถูกผ่านการวิเคราะห์และการสร้างเป็นดัชนีแต่จะไม่ถูกจัดเก็บในฐานข้อมูล เหมาะสำหรับ field ที่มีขนาดใหญ่และผู้ใช้ต้องการให้ระบบค้นหาแต่ไม่จะเป็นต้องแสดงผลในรูปแบบดั้งเดิม เช่น เอกสารบนเว็บ หรือเอกสารขนาดใหญ่ เป็นต้น
- **Text**: จะถูกผ่านการวิเคราะห์และการสร้างเป็นดัชนี รวมทั้งอาจจะยังถูกจัดเก็บในฐานข้อมูลด้วย ผู้ใช้ควรระวังในการใช้งาน คือถ้าลักษณะของข้อมูลที่ส่งไปใน field นี้เป็นแบบ String ข้อมูลนั้นจะถูกจัดเก็บในฐานข้อมูล แต่ถ้าข้อมูลเป็นแบบ Reader ก็จะไม่ถูกจัดเก็บในฐานข้อมูล

Field type	Analyzed	Indexed	Stored	Example usage
Field.Keyword (String, String) Field.Keyword (String, Date)		✓	✓	Telephone number and ID number, URLs, personal names, Dates
Field.UnIndexed (String, String)			✓	Document type (PDF, HTML, etc), used as a search criteria
Field.UnStored (String, String)	✓	✓		Document titles and content
Field.Text (String, String)	✓	✓	✓	Document titles and content
Field.Text (String, Reader)	✓	✓		Document titles and content

ตารางที่ 1: สรุปลักษณะและตัวอย่างการใช้งานของ Field แต่ละชนิด [ref. 2]

การเรียกใช้งาน Document และ Field:

Document และ Field มักจะถูกใช้ร่วมกัน การใช้งานสามารถทำได้โดยประกาศ Document เป็น instance ก่อน: เช่น ถ้าผู้ใช้มีเอกสารซึ่งสามารถจัดเป็น field ต่างๆ ดังนี้

title: Search Engine Performance
keyword: Internet
category: computer
content: Search engines are evaluated based on key performance measures: indexing and searching time.

สมมุติว่าเรากำหนดลักษณะของ Field ดังนี้ คือ title เป็นแบบ Text, keyword เป็นแบบ Keyword, category เป็นแบบ UnIndexed และ content เป็นแบบ UnStored การเขียนโปรแกรมจะสามารถทำได้ดังต่อไปนี้

```
Document doc = new Document();
doc.add(Field.Text("content", "Search Engine Performance"));
doc.add(Field.Keyword("keyword", "Internet"));
doc.add(Field.UnIndexed("category", "computer"));
doc.add(Field.UnStored("content", "Search engines are evaluated based on ... "));
```

2.2 Index API

Index API ซึ่งใช้ในการสร้างดัชนีนั้นมี class ที่เกี่ยวข้องดังนี้ คือ

- **IndexWriter**: เป็นส่วนประกอบหลักที่เกี่ยวข้องกับการสร้างดัชนี IndexWriter ทำหน้าที่สร้างและบันทึกดัชนีจากเอกสารที่ถูกแปลงให้เป็น instance ของ Document แล้ว ดังตัวอย่างต่อไปนี้

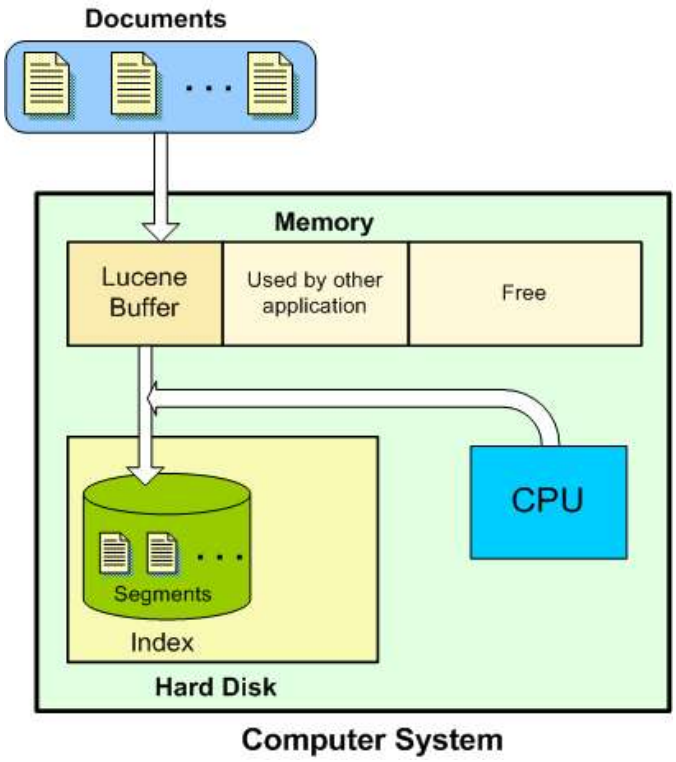
```
IndexWriter writer = new IndexWriter(directory, analyzer, true);
```

(**หมายเหตุ**: directory ระบุ location ที่เก็บฐานข้อมูลดัชนี analyzer ระบุวิธีที่ใช้วิเคราะห์ข้อความ (ดูรายละเอียดใน 2.4) `true` ระบุให้ลูซีนสร้างฐานข้อมูลดัชนีขึ้นมาใหม่แทนที่ดัชนีเดิม ถ้ามี)

หลังจากที่ผู้ใช้สร้างเอกสารแล้ว (จากตัวอย่างใน 2.1) สามารถสั่งให้ลูซีนสร้างดัชนี และเมื่อสิ้นสุดการสร้างดัชนีจากกลุ่มเอกสารแล้วต้องเรียกคำสั่งปิด ดังนี้

```
writer.addDocument(doc);
writer.close();
```

ฐานข้อมูลดัชนีที่ได้จากการใช้ IndexWriter จะเป็นลักษณะ segments ซึ่งเก็บอยู่ในหลายๆไฟล์ (รูปที่ 6) การออกแบบ IndexWriter นั้นมี hardware หลักที่เกี่ยวข้องคือ memory กับ hard disk ความเร็วในการทำงาน (Access Time) ของ memory กับ hard disk นั้นต่างกันมาก โดยที่ memory จะมีความเร็วที่วัดโดยหน่วย ns (nanosecond) และ hard disk จะมีความเร็วที่วัดโดยหน่วย ms (millisecond) เมื่อผู้ใช้ออกคำสั่งให้สร้างดัชนี ลูซีนจะยังไม่เขียนบันทึกดัชนีลงบน hard disk ในทันทีซึ่งจะทำให้ช้ามาก แต่จะรอรวบรวมดัชนีที่ได้จากหลายๆเอกสารก่อนบน memory แล้วจึงบันทึกดัชนีลงบน hard disk เป็นช่วงๆไป



<u>รูปที่ 6: กระบวนการสร้างดัชนีโดยลูซีน</u>

ในการสร้างดัชนีนั้นผู้ใช้สามารถปรับเปลี่ยนและควบคุมความเร็วในการสร้างดัชนีโดยระบุตัวแปรต่างๆคือ mergeFactor, maxMergeDocs และ minMergeDocs ตัวแปรเหล่านี้มีหน้าที่ในการควบคุมขนาดและจำนวนของ segments รวมทั้งความถี่ในการเขียนดัชนีลงบน hard disk เช่นถ้า mergeFactor เท่ากับ 100 หมายถึงจะมีการรวม segments เมื่อมีจำนวนถึง 100 แล้วทำการบันทึกลงบน hard disk ดังนั้นถ้าต้องการให้กระบวนการสร้างดัชนีรวดเร็วผู้ใช้ควรตั้ง mergeFactor ไว้สูงซึ่งช่วยลดความถี่ในการบันทึกข้อมูลลงใน hard disk แต่การตั้งค่า mergeFactor ไว้สูงนั้นหมายถึงจำนวนข้อมูลที่อยู่บน memory จะมีมากตามไปด้วย ดังนั้นระบบที่ใช้ต้องมีขนาด memory มากพอที่จะรองรับการทำงานได้ ผู้ใช้ยังสามารถระบุขนาดของ segment ได้โดยการตั้งค่า maxMergeDocs เช่น ถ้า maxMergeDocs เท่ากับ 1,000 หมายถึงแต่ละ segment สามารถมีจำนวน Document ได้มากที่สุดเท่ากับ 1,000 ส่วน minMergeDocs นั้นใช้สำหรับระบุจำนวน Document ขั้นต่ำก่อนที่จะเขียนลงใน segment ดังนั้นถ้าต้องการให้กระบวนการสร้างดัชนีรวดเร็วผู้ใช้ควรตั้ง minMergeDocs ไว้สูงแต่ขนาดของ memory ต้องใหญ่ตามไปด้วย แต่จะสังเกตได้ว่า minMergeDocs นั้นไม่มีผลต่อขนาดของ segment

เนื่องจากจำนวน segment ที่ได้จากการสร้างดัชนีนั้นอาจจะมีจำนวนมากซึ่งขึ้นอยู่กับค่า mergeFactor ในขั้นนี้ผู้

ใช้ก็สามารถที่จะทำการค้นหาจากกลุ่ม segments นี้ได้ แต่ถ้าจำนวน segment มีมากจะทำให้การค้นหาช้าตามไปด้วย เนื่องจากต้องค้นหาดัชนีในทุกๆ segment นั่นเอง ดังนั้นใน IndexWriter จะมีคำสั่ง optimize เพื่อรวมไฟล์ segments เข้าด้วยกันเป็นหนึ่ง segment การใช้งานคำสั่ง optimize สามารถทำได้ดังนี้

```
writer.optimize();
```

ผู้ใช้ควรระวังในการใช้คำสั่งนี้เนื่องจากต้องมีการใช้เรียกและเขียนข้อมูลบน hard disk (disk I/O activity) อย่างมากอาจทำให้การทำงานของระบบโดยรวมช้าและยังต้องอาศัยเนื้อที่ของ memory อย่างมากอีกด้วย การใช้คำสั่ง optimize นั้นช่วยให้กระบวนการค้นหาเร็วขึ้นเท่านั้นโดยที่ไม่มีผลต่อกระบวนการสร้างดัชนี

- **Directory**: มีหน้าที่ในการระบุตำแหน่งพื้นที่ที่ใช้จัดเก็บฐานข้อมูลดัชนี ในการพัฒนาระบบทั่วไปซึ่งมีปริมาณเอกสารมากนั้น ส่วนใหญ่จะต้องจัดเก็บใน hard disk ซึ่งต้องใช้ class FSDirectory ตามตัวอย่างต่อไปนี้

  - สร้างดัชนี:
    ```
    File indexDir = new File("dirName");
    Directory fsDir = FSDirectory.getDirectory(indexDir, true);
    IndexWriter writer = new IndexWriter(fsDir, analyzer, true);
    ```
  - ค้นคืนข้อมูล:
    ```
    File indexDir = new File("dirName");
    Directory fsDir = FSDirectory.getDirectory(indexDir, true);
    IndexSearcher searcher = new IndexSearcher(fsDir);
    ```

  นอกจาก FSDirectory แล้วยังมีการใช้งาน Directory อีกวิธีหนึ่งคือ RAMDirectory ซึ่งจัดการกับฐานข้อมูลดัชนีบน memory ไม่ว่าจะเป็นการเขียนดัชนีหรือการค้นหาก็ตาม ทำให้กระบวนการทุกอย่างเป็นไปอย่างรวดเร็ว เหมาะกับปริมาณเอกสารที่ไม่มากนัก และใช้สำหรับในการทดสอบระบบหรือวิธีต่างๆที่ต้องการดูผลลัพธ์อย่างรวดเร็ว การใช้งานของ RamDirectory จะมีการเรียกเหมือนกับ FSDirectory ทุกประการ

2.3 Search API

Search API ซึ่งใช้ในส่วนของการค้นคืนนั้นมี class ที่เกี่ยวข้องดังนี้ คือ

- **IndexSearcher**: เป็นส่วนประกอบหลักที่เกี่ยวข้องกับการค้นคืนข้อมูลจากดัชนี IndexSearcher ทำหน้าที่เปิดฐานข้อมูลดัชนีในโหมดอ่านเท่านั้น (Read-only) และค้นคืนเอกสารโดยดูจากคิวรี่ของผู้ใช้ การใช้งาน IndexSearcher สามารถทำได้โดยประกาศให้เป็น instance เสียก่อน โดยสามารถระบุโดยใช้ Directory หรือ โดยใส่ที่อยู่ของไฟล์ดัชนี (File system path) แต่แนะนำให้ใช้แบบแรกเนื่องจากมีความยืดหยุ่นในการใช้งานมากกว่าเช่น อาจจะค้นหาจากดัชนีที่อยู่บน memory ก็ได้ ตัวอย่างการเรียกใช้งานผ่าน FSDirectory สามารถทำได้ดังนี้

```
Directory directory = FSDirectory.getDirectory(indexDir, false);
IndexSearcher is = new IndexSearcher(directory);
```

ค่าที่ตั้งไว้ใน FSDirectory เป็น false หมายถึงเป็นการเปิดดัชนีที่สร้างไว้แล้วไม่ใช่ต้องการสร้างขึ้นมาใหม่ วิธีการค้นหามี 3 แบบ แต่โดยทั่วไปแล้วนิยมใช้ในรูปแบบต่อไปนี้ `Hits search(Query q)` ซึ่งเป็นการค้นหาผ่าน class Query ที่ประกอบด้วยคำต่างๆที่ผู้ใช้ต้องการค้นหา

- **Term**: เป็นหน่วยย่อยสำหรับใช้ในการค้นคืน Term มีลักษณะคล้ายกับ Field คือประกอบด้วยชื่อ field และค่าใน field นั้น การใช้งาน Term คู่กับ TermQuery สามารถทำได้ดังตัวอย่างต่อไปนี้

```
Query q = new TermQuery(new Term("contents", "lucene"));
Hits hits = is.search(q);
```

ตัวอย่างนี้เป็นการสั่งให้ลูซีนหาเอกสารทั้งหมดที่มีคำว่า "lucene" ปรากฏอยู่ใน field "contents"

- **Query**: เป็น class ซึ่งทำหน้าที่สร้างคิวรี่หรือกลุ่มของคำพร้อมทั้งโอเปอเรชั่น (Operation) ที่เกี่ยวข้องเช่น เครื่องหมายตรรกะ เป็นต้น การใช้งาน Query มีอยู่หลากหลายรูปแบบด้วยกัน โดยจะยกตัวอย่างรูปแบบที่มีการใช้งานบ่อยดังนี้ คือ
  - **TermQuery**: เป็นการค้นหาที่ง่ายที่สุดโดยดูคำที่ปรากฏใน field ตัวอย่างการใช้งานคือ

```
Term t = new Term("contents", "java");
Query query = new TermQuery(t);
```

ตัวอย่างนี้เป็นการสั่งให้ลูซีนหาเอกสารทั้งหมดที่มีคำว่า "java" ปรากฏอยู่ใน field "contents" การใช้งาน TermQuery มีประโยชน์อย่างมากในกรณีที่ต้องการค้นหาเอกสารจาก key อย่างในกรณีของการใช้ Field.Keyword() ตัวอย่างเช่น การค้นหาโดยใช้รหัส ISBN ของหนังสือ

```
Term t = new Term("isbn", "1930110995");
Query query = new TermQuery(t);
```

  - **RangeQuery**: เป็นการค้นหาจากช่วงของดัชนี เนื่องจากฐานข้อมูลดัชนีนั้นมีการจัดเรียงแบบตามตัวอักษร การใช้ RangeQuery สามารถทำได้ไม่ยากโดยระบุค่าเริ่มต้นและค่าสิ้นสุด ตัวอย่างการใช้งานมีดังต่อไปนี้

```
Term begin = new Term("year", "1988");
Term end = new Term("year", "1990");
RangeQuery query = new RangeQuery(begin, end, true);
```

  - **PrefixQuery**: เป็นการค้นหาเอกสารทั้งหมดที่ขึ้นต้นด้วยคำหรือข้อความที่ระบุในคิวรี่ ตัวอย่างการใช้งานเช่นค้นหา path ของไดเร็กทอรี่ หรือหมวดหมู่ของเอกสารที่เป็นโครงสร้าง ดังต่อไปนี้

```
Term begin = new Term("category", "/technology/computer/programming");
PrefixQuery query = new PrefixQuery(term);
```

  - **BooleanQuery**: เป็นการค้นคืนจากกลุ่มของคำโดยมีเครื่องหมายตรรกะเป็นตัวเชื่อมซึ่งได้แก่ AND, OR, NOT รูปแบบการใช้งาน BooleanQuery มีดังนี้

    public void add(Query query, boolean required, boolean prohibited)

    การตั้งค่าของ required และ prohibited เป็นการกำหนดฟังก์ชั่นของตรรกะซึ่งมีได้ 3 แบบ คือ

    - required=false, prohibited=false: หมายถึงข้อความนั้นจะมีหรือไม่มีในเอกสารก็ได้ (OR)
    - required=false, prohibited=true: หมายถึงข้อความนั้นจะต้องไม่มีในเอกสาร (NOT)

- required=true, prohibited=false: หมายถึงข้อความนั้นจะต้องมีในเอกสาร (AND)

ตัวอย่างการใช้งาน BooleanQuery ในการค้นหารายชื่อหนังสือมีดังต่อไปนี้

```
TermQuery subjectq = new TermQuery(new Term("subject", "search"));
RangeQuery yearq = new RangeQuery(new Term("year", "2003"),
                                  new Term("year", "2004"), true);
BooleanQuery bq = new BooleanQuery();
bq.add(subjectq, true, false);
bq.add(yearq, true, false);
```

ตัวอย่างข้างต้นนี้เป็นการค้นหาหนังสือที่มีคำว่า "search" ใน field "subject" และตีพิมพ์ในระหว่างปี 2003-2004 (เป็นการทำตรรกะแบบ AND)

- **PhraseQuery**: เป็นวิธีการค้นหาแบบกลุ่มของคำโดยผู้ใช้สามารถระบุระยะห่างของคำได้ (เรียกว่า slop) ตัวอย่างการใช้งาน PhraseQuery ในการค้นหาจากประโยค "A dog is running on the street." มีดังต่อไปนี้

```
PhraseQuery q = new PhraseQuery();
q.setSlop(4);
q.add(new Term("field", "dog");
q.add(new Term("field", "street");
```

จากตัวอย่างการใช้ PhraseQuery ก็จะสามารถค้นหาประโยคข้างต้นได้พบเนื่องจากเราได้กำหนดระยะห่าง(slop) ไว้มากที่สุดเท่ากับ 4

- **WildcardQuery**: เป็นวิธีการค้นหาโดยใช้ส่วนหนึ่งของคำหรือวลี ในลูซีนผู้ใช้สามารถกำหนด WildCardQuery ได้ 2 รูปแบบคือ ใช้ * แทนตัวอักษรตั้งแต่ศูนย์ตัวขึ้นไปและ ? แทนตัวอักษรตั้งแต่ศูนย์หรือหนึ่งตัว ตัวอย่างการใช้งาน WildcardQuery มีดังต่อไปนี้

```
Query query = new WildcardQuery(new Term("contents", "?old*"));
```

คิวรี่ข้างต้นนี้จะค้นคืนเอกสารที่มีตัวอักษรศูนย์หรือหนึ่งตัวนำหน้าและตัวอักษรศูนย์หรือมากกว่าตามหลัง "old" ดังนั้นเอกสารที่มีคำเช่นว่า "old", "gold", "sold", "golden" ก็จะถูกค้นคืนมาด้วย

- **FuzzyQuery**: เป็นวิธีการค้นหาคำที่ใกล้เคียงกับคำที่ระบุในคิวรี่โดยอาศัยการคำนวณระยะความแตกต่างของตัวอักษร (Edit distance) เช่นระยะความแตกต่างของคำ "three" และ "tree" เท่ากับ 1 เป็นต้น

- **QueryParser**: มีหน้าที่ในการประมวลผลและแปลงข้อความคิวรี่(Query expression) ที่ผู้ใช้ส่งให้แก่ระบบให้อยู่ในรูปของ Query คิวรี่ที่สร้างโดยผู้ใช้อาจจะมีความซับซ้อนมากเช่น

+pubdate:[20040101 TO 20041231] Java AND (Jakarta OR Apache)

เป็นคิวรี่ที่ใช้สำหรับค้นหารายชื่อหนังสือทั้งหมดที่มีคำว่า "Java" และ "Jakarta" หรือ "Apache" และตีพิมพ์ในปี 2004 การประมวลคิวรี่สามารถใช้ shortcut syntax ได้ดังนี้

- a AND b: สามารถเขียนได้เป็น +a +b
- a OR b: สามารถเขียนได้เป็น a b
- a AND NOT b: สามารถเขียนได้เป็น +a -b

การใช้งาน QueryParser ต้องมีการระบุ Analyzer ที่ใช้ในการวิเคราะห์โดยทั่วไปจะใช้แบบเดียวกับที่ใช้ตอนสร้างดัชนี รูปแบบการใช้งานมีดังต่อไปนี้

```
Query query = QueryParser.parse("This is some sentence.", "contents", new
StandardAnalyzer());
```

<u>การใช้งาน QueryParser สำหรับค้นคืนเอกสารภาษาไทย:</u>

ในขั้นตอนของการประมวลผลข้อความไม่ว่าจะในระหว่างการสร้างดัชนีหรือการค้นคืนก็ตาม จะต้องมีการใช้ Analyzer ให้เหมาะกับลักษณะของข้อความซึ่งรวมถึงภาษาที่ใช้ สำหรับภาษาไทยนั้น เนื่องจากภาษาไทยมีการเขียนที่ต่อเนื่องโดยไม่มีการเว้นวรรคตอนที่แน่นอน ทางทีมสรรสารได้พัฒนา ThaiAnalyzer ซึ่งสามารถวิเคราะห์ได้ทั้งภาษาไทยและภาษาอังกฤษ รายละเอียดของ ThaiAnalyzer อยู่ในส่วน 2.4 ซึ่งจะมีตัวอย่างในการใช้ ThaiAnalyzer ในการสร้างดัชนี รวมทั้งมีการเปรียบเทียบผลที่ได้จากการใช้ Analyzer แบบต่างๆ

ในกรณีทั่วไปการสร้างดัชนีหรือการค้นคืนก็ควรจะใช้ Analyzer แบบเดียวกัน ตัวอย่างต่อไปนี้จะแสดงให้เห็นถึงการใช้ ThaiAnalyzer ในการวิเคราะห์คิวรี่สำหรับ QueryParser รวมทั้งการใช้ Query ที่เหมาะกับภาษาไทย

<u>ตัวอย่างข้อความ</u>: ข้อความที่ใช้ในการทดสอบมี 3 ข้อความ ซึ่งเมื่อใช้ ThaiAnalyzer จะมีการแบ่งคำ (Tokenizing) และส่งไปทำการสร้างดัชนี ดังนี้

ข้อความที่ 1: รัฐบาลไทยร่วมสนับสนุนการท่องเที่ยวในประเทศ
[รัฐบาล] [ไทย] [ร่วม] [สนับสนุน] [การ] [ท่องเที่ยว] [ใน] [ประเทศ]
ข้อความที่ 2: เที่ยวทั่วไทยในแบบอเมซิ่งไทยแลนด์
[เที่ยว] [ทั่ว] [ไทย] [ใน] [แบบ] [อ] [เม] [ซิ่งไทย] [แลนด์]
ข้อความที่ 3: นักท่องเที่ยวจากต่างประเทศเดินทางมาประเทศไทย
[นักท่องเที่ยว] [จาก] [ต่าง] [ประเทศ] [เดินทาง] [มา] [ประเทศ] [ไทย]

<u>ข้อสังเกต :</u>

- การตัดคำแม้จะเป็นคำเดียวกันแต่ถ้าปรากฏอยู่ในข้อความแวดล้อมที่ต่างกันก็อาจจะให้ผลที่ต่างกันได้เช่น คำว่า "ท่องเที่ยว" ในข้อความที่ 1 กับในข้อความที่ 2
- การตัดคำสำหรับคำที่ไม่ปรากฏในพจนานุกรม (Unknown word) อาจจะมีความผิดพลาดได้เช่น คำว่า "อเมซิ่ง" ซึ่งอาจจะทำให้การตัดคำผิดพลาดไปถึงคำที่อยู่ถัดไป

ตัวอย่างต่อไปนี้จะสาธิตการค้นคืนข้อความโดยใช้คำต่างๆทั้งในรูปแบบของคำเดี่ยวและกลุ่มของคำที่อยู่ในรูปแบบข้อความตรรกะ (Boolean expression)

<u>ตัวอย่างการใช้ QueryParser โดยผ่าน ThaiAnalyzer</u>:

- **กรณีที่ 1:** คิวรี่ที่เป็นส่วนหนึ่งของคำ
  ลักษณะของคิวรี่: ถ้ามีคำว่า "abcd" และคิวรี่คือ "ab" "abc" "bc" "bcd" "cd"
  ตัวอย่างคิวรี่: เที่ยว
  ผลลัพธ์: เมื่อค้นคืนจะได้ข้อความที่ 2 เท่านั้น แม้ว่าข้อความที่ 1 จะมีคำว่า "เที่ยว" อยู่ แต่การตัดคำใน ThaiAnalyzer จะได้คำว่า "ท่องเที่ยว" ดังนั้นจึงทำให้ไม่พบเอกสารนี้ และในข้อความที่ 3 แม้จะมีคำว่า "เที่ยว" แต่การตัดคำจะได้คำว่า "นักท่องเที่ยว" ดังนั้นจึงทำให้ไม่พบเอกสารนี้เช่นกัน

- **กรณีที่ 2:** คิวรี่ที่ประกอบด้วยของกลุ่มคำที่มีการแบ่งขอบเขตที่ชัดเจน
  ลักษณะของคิวรี่: ถ้ามีคำ 2 คำได้แก่ "ab" และ "cd" และคิวรี่คือ "abcd"
  ตัวอย่างคิวรี่: ประเทศไทย
  ผลลัพธ์: เมื่อค้นคืนจะได้ข้อความที่ 3 อย่างถูกต้อง เนื่องจาก QueryParser จะทำการคัดคำและค้นคืนตามกลุ่มของคำในลักษณะ PhraseQuery ดังนั้นจึงไม่เกิดปัญหาใดๆ

- **กรณีที่ 3:** คิวรี่ที่ประกอบด้วยกลุ่มคำทั้งที่มีขอบเขตชัดเจนและที่เป็นส่วนหนึ่งของคำอื่น
  ลักษณะของคิวรี่: ถ้ามีคำ 2 คำได้แก่ "ab" และ "cdef" และคิวรี่คือ "abcd" หรือ
  ถ้ามีคำ 2 คำได้แก่ "abcd" และ "ef" และคิวรี่คือ "cdef"
  ตัวอย่างคิวรี่: ไทยแลนด์
  ผลลัพธ์: เมื่อค้นคืนจะไม่พบข้อความใดๆ

ดังนั้นคำถามที่ตามมาคือจะมีวิธีแก้ไขปัญหานี้ได้อย่างไรจึงจะมั่นใจว่าผลลัพธ์ที่ได้จะมีความถูกต้อง (Precision) และมีความครบถ้วน (Recall) สูง คำตอบคือต้องมีการใช้เทคนิคการแปลงคิวรี่จากผู้ใช้ให้อยู่ในแบบที่ให้ผลลัพธ์ตามที่เราต้องการเช่นในกรณีปัญหาข้างต้นนี้

- **วิธีแก้ไขกรณีที่ 1**: ถ้าคิวรี่เป็นส่วนหนึ่งของคำ ก็สามารถแก้ไขได้ไม่ยากโดยการใช้ WildcardQuery จากตัวอย่างข้างต้น เราสามารถใช้ WildcardQuery ในการค้นหาคำว่า "เที่ยว" ได้ดังนี้

```
IndexSearcher is = new IndexSearcher(indexDir);
Query query = new WildcardQuery(new Term("content", "*เที่ยว*"));
Hits hits=is.search(query);
```

ผลลัพธ์ที่ได้คือข้อความ 1, 2 และ 3 ตามลำดับ

- **วิธีแก้ไขกรณีที่ 3**: สามารถแก้ไขได้โดยตัดคำโดยใช้ ThaiAnalyzer ก่อนแล้วจึงนำไปสร้างเป็นคิวรี่โดยใช้ WildcardQuery กับแต่ละคำที่ตัดออกมาแล้วประสมกันโดยใช้ BooleanQuery จากตัวอย่างข้างต้นเราสามารถค้นหาคำว่า "ไทยแลนด์" ได้ดังนี้

```
IndexSearcher is = new IndexSearcher(indexDir);
```

```
ThaiAnalyzer analyzer=new ThaiAnalyzer();
BooleanQuery bq = new BooleanQuery();
TokenStream stream = analyzer.tokenStream("contents",
                                          new StringReader(str));
while (true) {
  Token token = stream.next();
  if (token == null) break;
  Query q = new WildcardQuery(new Term("content",
                                       "*" + token.termText() + "*"));
  bq.add(q, true, false);
}
Hits hits=searcher.search(bq);
```

จะเห็นได้ว่าเราสามารถใช้โปรแกรมที่มีอยู่ในตัวลูซีนเองในการแก้ไขปัญหาที่เกิดขึ้นในการค้นคืนข้อมูลที่เป็นภาษาไทย เพื่อให้การใช้งานในส่วนค้นคืนเป็นไปอย่างง่ายและสะดวก ทีมสรรสารได้พัฒนาโปรแกรมในลักษณะโมดูลซึ่งผู้พัฒนาระบบสามารถเรียกใช้ในการค้นคืนข้อมูลภาษาไทยโดยไม่จำเป็นต้องรู้รายละเอียดเกี่ยวกับ QueryParser และเทคนิคการแก้ไขปัญหาที่กล่าวมาในข้างต้นนี้ โดยโมดูลนี้อยู่ใน Searcher API ซึ่งเป็นส่วนหนึ่งของโปรแกรม สรรสาร ลุ๊ค (Sansarn Look!) (ดูรายละเอียดในหัวข้อที่ 5)

- **Hits**: เป็นส่วนของผลลัพธ์ที่ได้จากการค้นคืน ซึ่งได้มาจากการเรียก search ใน IndexSearcher ฟังก์ชั่นใน Hits มีอยู่ 4 อย่างคือ
  - length(): จำนวนเอกสารที่ได้จากการค้นคืน (จำนวน hits)
  - doc(n): เอกสารลำดับที่ n จากผลลัพธ์ที่ค้นคืนมาได้ทั้งหมด (เป็น instance ของ Document)
  - id(n): หมายเลขของเอกสาร (ID) ของเอกสารลำดับที่ n ของผลลัพธ์
  - score(n): คะแนนของเอกสารที่ปรับค่าแบบ normalized โดยเอาคะแนนของเอกสารอันดับแรกเป็นหลัก คะแนนจะมีค่ามากกว่า 0 และน้อยกว่าหรือเท่ากับ 1

การใช้ Hits จะมีการเก็บเอกสารไว้ 100 เอกสารไว้ใน cache เนื่องจากโดยทั่วไปแล้วผู้ใช้จะต้องการดูผลลัพธ์ในช่วงแรกเท่านั้น ในกรณีที่ผลลัพธ์มีจำนวนยาวมาก การแสดงผลจากการค้นคืนในแต่ละหน้าจะสามารถทำได้ 2 วิธี คือ

- ประกาศ IndexSearcher และ Hits เป็น instance และคงค้างไว้สำหรับแต่ละผู้ใช้
- ทำการค้นคืนใหม่ทุกครั้งที่ผู้ใช้ต้องการผลในหน้าต่อไป (Requery) วิธีนี้เป็นวิธีที่เหมาะกับการใช้งานโดยเฉพาะสำหรับ Web application ที่มีการใช้งานของ HTTP ซึ่งเป็นแบบไม่มีการเก็บสถานะของผู้ใช้ (Stateless)

2.4 Analysis API

ส่วนนี้มีหน้าที่หลักในการแปลงข้อความในเอกสาร (Text) ให้เป็นรายการของคำ (Index Terms) ซึ่งเป็นหน่วยย่อยที่สุดสำหรับนำไปสร้างดัชนี ในลูซีน class ที่ทำหน้าที่ในส่วนนี้คือ Analyzer ซึ่งมีอยู่หลายชนิด แตกต่างกันไปในเทคนิคที่ใช้วิเคราะห์ข้อความ เช่น การลบเอาตัวอักขระพิเศษต่างๆออก การเปลี่ยนอักษรตัวใหญ่เป็นตัวเล็ก (Normalizing) การแปลงคำให้อยู่ในรูปรากศัพท์ (Stemming) การตัดคำที่ไม่สำคัญออกได้ (Stopword Removal) เป็นต้น โดยรวมแล้วก็คือการแปลงข้อความให้เป็นคำย่อยๆ ซึ่งมีชื่อเรียกว่า Tokenization หน่วยของคำนี้เรียกว่า Tokens ซึ่งเมื่อนำมาผนวกกับ field แล้วก็เป็น Terms

การเลือกชนิดของ Analyzer เป็นสิ่งที่สำคัญเป็นอย่างมากในการพัฒนาระบบ ถ้าใช้ Analyzer ที่ไม่เหมาะสมก็อาจจะเป็นสาเหตุให้ไม่สามารถค้นคืนเอกสารได้อย่างถูกต้องและแม่นยำ โดยเฉพาะอย่างยิ่งในกรณีของข้อความที่ไม่ใช่ภาษาอังกฤษ Analyzer ที่มีมากับลูซีนนั้นมีอยู่ 4 ชนิด ได้แก่

- **WhiteSpaceAnalyzer**: แบ่ง tokens ตาม whitespace ซึ่งได้แก่ ตัวอักษรที่ใช่แบ่งช่องว่าง เช่น space, tab, new-line characters เป็นต้น
- **SimpleAnalyzer**: แบ่ง tokens ตามตัวอักขระพิเศษที่ไม่ใช่ตัวอักษร เช่น semi-colon, period, @ เป็นต้น รวมทั้งเปลี่ยนอักษรตัวใหญ่เป็นตัวเล็กทั้งหมด
- **StopAnalyzer**: คล้ายกับ SimpleAnalyzer คือ แบ่ง tokens ตามตัวอักขระพิเศษที่ไม่ใช่ตัวอักษร เช่น semi-colon, period, @ เป็นต้น รวมทั้งเปลี่ยนอักษรตัวใหญ่เป็นตัวเล็กทั้งหมด แต่ยังรวมถึงการตัดคำที่ไม่สำคัญในภาษาอังกฤษออก (Stopword Removal)เช่น "a" "an" "and" "but" "if" เป็นต้น
- **StandardAnalyzer**: คล้ายกับ StopwordAnalyzer แต่ยังมีการวิเคราะห์หลักไวยากรณ์เพิ่มเติม เช่น สามารถรู้ลักษณะของที่อยู่ e-mail และ ที่อยู่ของเว็บไซต์ เป็นต้น

โดยทั่วไปแล้วถ้าเอกสารอยู่ในภาษาอังกฤษทั้งหมดชนิดของ Analyzer ที่เหมาะสมกับการใช้งานในกรณีทั่วไปคือ StandardAnalyzer แต่ถ้าเป็นเอกสารทั้งภาษาอังกฤษและไทยแล้ว การใช้ StandardAnalyzer จะทำได้ไม่ดี แม้ว่าได้มีนักพัฒนาโปรแกรมได้ต่อยอดโดยการทำในส่วนของภาษาเยอรมันและภาษาจีนแล้ว แต่ยังไม่มีการพัฒนาในส่วนของภาษาไทย ดังนั้นทีมสรรสารซึ่งมีความรู้และความเชี่ยวชาญในการวิเคราะห์และประมวลผลภาษาไทยจึงได้ทำการพัฒนาต่อยอดในส่วนของการวิเคราะห์ภาษาไทย (ThaiAnalyzer) สำหรับการใช้งานลูซีนโดยเฉพาะ ซึ่งทำให้การค้นคืนเอกสารที่มีทั้งภาษาไทยและอังกฤษเป็นไปอย่างมีประสิทธิภาพมากยิ่งขึ้น

- **ThaiAnalyzer**: เป็นการต่อยอดจาก StopwordAnalyzer ซึ่งเหมาะกับภาษาอังกฤษ แต่ ThaiAnalyzer มีการตรวจสอบว่าข้อความนั้นมีส่วนที่เป็นภาษาไทยหรือไม่ ถ้ามี ThaiAnalyzer จะทำการตัดคำจากข้อความภาษาไทยนั้น

<u>การเรียกใช้งาน Analyzer แบบต่างๆ:</u>

ในการพัฒนาระบบค้นคืนข้อมูล Analyzer จะถูกใช้ใน 2 ส่วนด้วยกัน คือ

1) **การสร้างดัชนี**: ผู้ใช้ต้องระบุชนิดของ Analyzer ที่จะใช้ในการทำ Tokenizing เช่น ถ้าใช้ StandardAnalyzer ก็มีการเขียนโปรแกรมดังนี้

```
Analyzer analyzer = new StandardAnalyzer();
IndexWriter writer = new IndexWriter(directory, analyzer, true);
```

ชนิดของ Field ที่จะสามารถทำการ Tokenizing มี 2 แบบคือ Text และ Unstored ตัวอย่างต่อไปนี้แสดงการสร้างดัชนีจาก Field ทั้งสองแบบ

```
Document doc = new Document();
doc.add(Field.Text("title", "This is title."));
doc.add(Field.Unstored("contents", "...document contents..."));
writer.addDocument(doc);
```

ถ้าหากผู้ใช้ต้องการใช้ Analyzer ที่แตกต่างสำหรับแต่ละเอกสาร ก็สามารถระบุชนิดได้ในขั้นตอนการทำดัชนีดังนี้

```
writer.addDocument(doc, analyzer);
```

2) **การวิเคราะห์คิวรี่**: ผู้ใช้ต้องระบุชนิดของ Analyzer ที่จะใช้ในการวิเคราะห์คิวรี่ โดยที่ชนิดของ Analyzer ควรจะเป็นตัวเดียวกันกับที่ใช้ในการสร้างดัชนี การใช้คิวรี่ในการค้นคืนเอกสารสามารถทำได้สองรูปแบบ ดังนี้

- เรียกใช้แบบ static:

```
Query query = QueryParser.parse(expression, "contents", analyzer);
```

- เรียกโดยประกาศ QueryParser เป็น instance ก่อน:

```
QueryParser parser = new QueryParser("contents", analyzer);
query = parser.parse(expression);
```

<u>เปรียบเทียบ Analyzer แบบต่างๆ:</u>

ทางทีมสรรสารได้ทำการเปรียบเทียบผลลัพธ์ที่ได้จากการทำ tokenizing โดยใช้ Analyzer แบบต่างๆ ดังนี้

<u>ตัวอย่าง 1</u>: ข้อความทดสอบคือ "The XY&Z Corporation: - xyz@example.com"
ผลการทดสอบมีดังนี้

- **WhitespaceAnalyzer**: [The] [XY&Z] [Corporation:] [-] [xyz@example.com]
- **SimpleAnalyzer**: [the] [xy] [z] [corporation] [xyz] [example] [com]
- **StopAnalyzer**: [xy] [z] [corporation] [xyz] [example] [com]
- **StandardAnalyzer**: [xy&z] [corporation] [xyz@example.com]
- **ThaiAnalyzer**: [xy] [z] [corporation] [xyz] [example.com]

จากผลที่ได้จะเห็นได้ว่า Analyzer มีความสามารถในการวิเคราะห์ข้อความที่ต่างกันซึ่งจะขึ้นกับการใช้งานเป็นหลัก

<u>ตัวอย่าง 2</u>: ข้อความทดสอบคือ "กทม.ร่วมสนับสนุนการท่องเที่ยว Amazing Thailand 2005"
ผลการทดสอบมีดังนี้

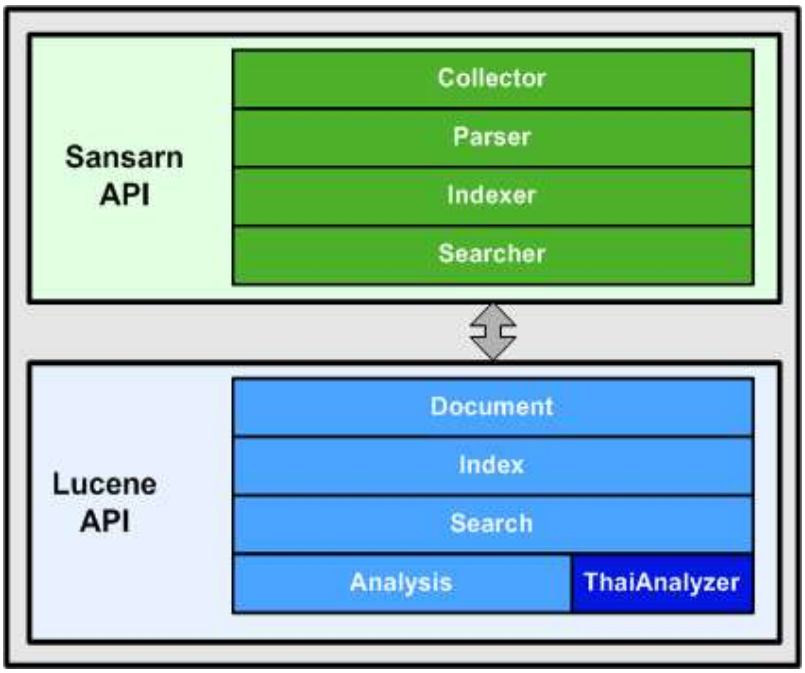
- **WhitespaceAnalyzer**: [กทม.ร่วมสนับสนุนการท่องเที่ยว] [Amazing] [Thailand] [(2005)]
- **SimpleAnalyzer**: [กทม] [ร] [วมสน] [บสน] [นการท] [องเท] [ยว] [amazing] [thailand]
- **StopAnalyzer**: [กทม] [ร] [วมสน] [บสน] [นการท] [องเท] [ยว] [amazing] [thailand]
- **StandardAnalyzer**: [กทม.ร่วมสนับสนุนการท่องเที่ยว] [amazing] [thailand] [2005]
- **ThaiAnalyzer**: [กทม] [ร่วม] [สนับสนุน] [การ] [ท่องเที่ยว] [amazing] [thailand] [2005]

จากผลที่ได้จะเห็นได้ว่า ThaiAnalyzer สามารถวิเคราะห์ข้อความที่เป็นภาษาไทยได้ดีกว่า Analyzer แบบอื่นๆ ทำให้ ThaiAnalyzer เหมาะกับการใช้งานสำหรับการค้นคืนเอกสารที่มีทั้งภาษาไทยและอังกฤษ นอกจากทำให้ผลการค้นคืนที่ถูกต้องและรวดเร็วกว่าแบบอื่นแล้วยังช่วยประหยัดเนื้อที่ในการจดเก็บฐานข้อมูลดัชนีอีกด้วย

<u>4. แนะนำโปรแกรมสรรสารลุ๊ค (Introduction to Sansarn Look!)</u>

เนื่องจากการพัฒนาระบบค้นคืนข้อมูลต่างๆจะมีฟังก์ชั่นการทำงานที่คล้ายกัน และเพื่อให้นักพัฒนาระบบและโปรแกรมเมอร์ทั่วๆไปสามารถนำเอาลูซีนไปใช้งานได้อย่างง่ายและสะดวกยิ่งขึ้น ทางทีมสรรสารอยู่ในระหว่างการพัฒนาชุดรหัสโปรแกรมซึ่งเรียกใช้รหัสโปรแกรมลูซีน ชุดรหัสโปรแกรมนี้มีชื่อว่า สรรสาร ลุ๊ค (Sansarn Look!) โครงสร้างทางสถาปัตยกรรมของสรรสาร ลุ๊ค อยู่ในรูปที่ 7 จะเห็นได้ว่าสรรสารอยู่ในชั้น (Layer) บนของลูซีน ในสรรสารนั้นมีรหัสโปรแกรมต่างๆในลักษณะโมดูลที่สามารถแบ่งตามหน้าที่ออกเป็น 4 ส่วนคือ

- **Collector:** มีหน้าที่เก็บเอกสารซึ่งอยู่ในรูปแบบ (Format) ต่างๆ เช่น HTML, PDF, DOC ทั้งที่อยู่ในเครื่อง (Local) และ ที่อยู่ห่างไกลออกไป (Remote) เช่นเอกสารบนเว็บผ่านทาง HTTP เป็นต้น
- **Parser:** มีหน้าที่หลักในการวิเคราะห์และสกัดเอาส่วนที่เป็นข้อความจากเอกสารในรูปแบบ (Format) ต่างๆ ทั้งนี้ยังรวมถึงการตัด tag ต่างๆออกจากเอกสารประเภท HTML และ XML
- **Indexer:** มีหน้าที่หลักในการสร้างดัชนีของคำจากเอกสารโดยการเรียกใช้โปรแกรมพื้นฐาน (Classes) ของลูซีน โปรแกรมในส่วน Indexer นี้ช่วยให้การสร้างดัชนีเป็นไปอย่างกึ่งอัตโนมัติคือ เริ่มจากผู้ใช้สร้างไฟล์ configuration ซึ่งกำหนด field ต่างและประเภทของดัชนีในแต่ละ field แล้วจึงสั่งให้โปรแกรมนี้ทำการสร้างดัชนีโดยอัตโนมัติต่อไป
- **Searcher:** มีหน้าที่หลักในการสร้างหน้าเว็บที่ใช้ในการค้นคืนซึ่งแบ่งออกเป็น 2 หน้าตามการใช้งานคือ หน้าเว็บสำหรับรับคำค้นคืนจากผู้ใช้ (Query Page) และหน้าเว็บสำหรับแสดงผลการค้นคืน (Search Result Page)



รูปที่ 7: โครงสร้างทางสถาปัตยกรรมของสรรสาร ลุ๊ค (Sansarn Look! API)

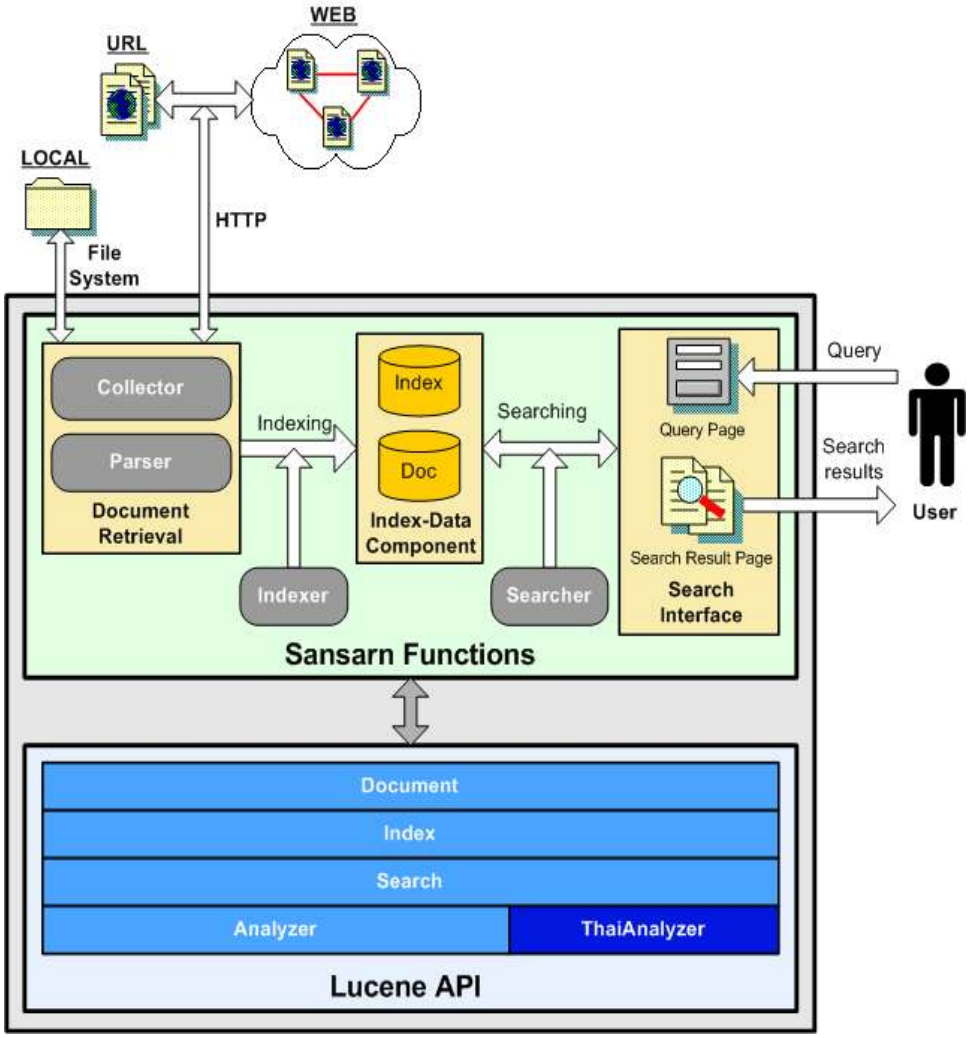
นักพัฒนาระบบสามารถสร้างระบบค้นคืนข้อมูลทั่วๆไปได้โดยการเรียกใช้โปรแกรมในส่วนต่างๆของสรรสาร (รูปที่ 8) เราสามารถแบ่งการทำงานของระบบค้นคืนข้อมูลทั่วไปสามารถแบ่งออกเป็น 2 ขั้นตอน คือ

1) **ขั้นตอนการสร้างดัชนี (Indexing Phase):** สรรสารรองรับข้อมูล 3 ประเภทด้วยกันคือ
   - เอกสารที่อยู่บนระบบไฟล์ของเครื่อง (Local): ไม่จำเป็นต้องโหลดเก็บเอกสาร สามารถสร้างดัชนีได้ทันที ซึ่งอาจจะต้องมีการแปลงรูปแบบไฟล์ก่อนโดยใช้โปรแกรมในส่วนของ Parser ที่เหมาะสม
   - เอกสารที่เก็บในเครื่องเซอร์เวอร์อื่น (URL): ในกรณีที่ต้องการสร้างดัชนีสำหรับค้นคืนเอกสารที่อยู่ห่างไกลออกไป ผู้ใช้ปลายทางก็สามารถเตรียมเอกสารในไดเร็กทอรี่สาธารณะ ดังนั้นเอกสารแต่ละฉบับจะมี URL กำกับไว้ และโปรแกรมในส่วนของ Collector ก็จะสามารถเข้าไปเก็นเอกสารได้โดยผ่าน HTTP
   - เอกสารบนเว็บ (Web): เป็นหน้าเว็ปทั่วไป (Web pages) ที่อยู่บนเครือข่ายอินเตอร์เน็ต เอกสารประเภทนี้เป็นลักษณะ Hypertext คือสามารถมี link ที่เชื่อมโยงไปยังเอกสารอื่นๆได้ ดังนั้นโปรแกรมในส่วนนี้จะมี crawler

หรือ web bot ซึ่งมีหน้าที่ตามเก็บเอกสารอื่นๆที่เชื่อมโยงจากเอกสารหนึ่งๆได้

โปรแกรมที่เกี่ยวข้องกับการสร้างดัชนีคือ Collector, Parser และ Indexer เมื่อเสร็จสิ้นกระบวนการสร้างดัชนีแล้ว สิ่งที่ได้คือ Index-Data Component ซึ่งประกอบไปด้วย 2 ส่วนคือ ฐานข้อมูลเอกสาร (Document) และฐานข้อมูลดัชนี (Index)

2) **ขั้นตอนการค้นคืนเอกสาร (Retrieval Phase):** โปรแกรมที่เกี่ยวข้องกับการค้นคืนเอกสารคือ Searcher โปรแกรมใน Searcher มี 2 ส่วนหลักคือ ส่วนที่สร้างหน้าสำหรับค้นคืน (Query page) และหน้าของผลลัพธ์ (Search result page) และส่วนที่ทำการวิเคราะห์คิวรี่และทำการค้นคืนเอกสารจากดัชนี



รูปที่ 8: โครงสร้างการทำงานของสรรสาร ลุ๊ค

ในขณะนี้ทีมสรรสารอยู่ในระหว่างการพัฒนาโปรแกรมสรรสาร ลุ๊ค นักพัฒนาระบบที่สนใจสามารถติดตามสถานะของโครงการนี้ได้จากเว็ปไซต์สรรสาร (http://sansarn.com)

<u>เอกสารอ้างอิง:</u>

1. The Lucene web site: http://jakarta.apache.org/lucene
2. E. Hatcher and O. Gospo, "Lucene in Action," Manning Publications, 2005.
3. R. Baeza-Yates and B. Ribeiro-Neto, "Modern Information Retrieval," ACM Press, Addison Wesley, 1999.
4. W. B. Frakes and R. Baeza-Yates, eds., "Information Retrieval: Data Structures & Algorithms," Prentice Hall, 1992.

หมายเหตุ: ปรับปรุงแก้ไขครั้งล่าสุดเมื่อ 3 ส.ค. 2548

ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ , สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ
สงวนลิขสิทธิ์ ตามพระราชบัญญัติลิขสิทธิ์ พ.ศ. 2537