กลยุทธ์ในการค้นหาบนระบบเพียร์-ทู-เพียร์ สำหรับคำค้นหาแบบหลายคีย์เวิร์ด

SEARCHING STRATEGY IN PEER-TO-PEER SYSTEM FOR MULTI-KEYWORD QUERY

ธนพล จิตนุพงศ์

TANAPON JITNUPONG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2557

KMITL-2014-IT-M-001-005

SEARCHING STRATEGY IN PEER-TO-PEER SYSTEM FOR MULTI-KEYWORD QUERY

TANAPON JITNUPONG

A THESIS SUBMITTED IN PATIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2014

KMITL-2014-IT-M-001-005

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ กลยุทธ์ในการค้นหาบนระบบเพียร์-ทู-เพียร์ สำหรับคำค้นหาแบบหลายคีย์เวิร์ด

Searching Strategy in Peer-to-Peer System for Multi-Keyword Query

นักศึกษา นายธนพล จิตนุพงศ์

รหัสประจำตัว ๕๒๖๖๐๔๑๐

ปริญญา วิทยาศาสตรมหาบัณฑิต

สาขาวิชา เทคโนโลยีสารสนเทศ

อาจารย์ที่ปรึกษาวิทยานิพนธ์ รองศาสตราจารย์ ดร.นพพร โชติกกำธร

คณะกรรมการสอบวิทยานิพนธ์	ลายมือชื่อ
รองศาสตราจารย์ ดร. โชติพัชร์ ภรณวลัย	
รองศาสตราจารย์ ดร.อนันต์ ผลเพิ่ม	
รองศาสตราจารย์ ดร.นพพร โชติกกำธร	
ดร.สุเมธ ประภาวัต	

วัน / เดือน/ ปี ที่สอบ วันอังคารที่ ๒๐ พฤษภาคม ๒๕๕๗ เวลา ๐๙.๓๐ น.

สถานที่สอบ ณ ห้อง ๓๓๓ ชั้น ๓ คณะเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศรับรองแล้ว

(รองศาสตราจารย์ ดร.จันทร์บูรณ์ สถิตวิริยวงศ์)

คณบดีคณะเทคโนโลยีสารสนเทศ

วันที่ ๓๐ **เดือน** พฤษภาคม **พ.ศ.** ๒๕๕๗

หัวข้อวิทยานิพนธ์	กลยุทธ์ในการค้นหาบนระบบเพียร์-ทู-เพียร์ สำหรับคำค้นหาแบบหลายคีย์เวิร์ด
นักศึกษา	นายธนพล จิตนุพงศ์
รหัสนักศึกษา	52660410
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
แขนงวิชา	เทคโนโลยีระบบสารสนเทศ
พ.ศ.	2557
อาจารย์ที่ปรึกษา	รศ.ดร.นพพร โชติกกำธร

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้ นำเสนอกลยุทธ์ในการค้นหาบนระบบเพียร์ทูเพียร์สำหรับคำค้นหาแบบหลายคีย์เวิร์ด การค้นหาแบบหลายคีย์เวิร์ดในระบบเพียร์ทูเพียร์แบบ DHT-based ที่อาศัยการจัดเก็บดัชนีแบบเทอมเดี่ยว ต้องมีการรวบรวมรายการเอกสารจากโหนดที่กระจายกันอยู่ ทำให้เกิดปริมาณข้อมูลในเครือข่ายสูง การจัดหมู่เทอมก่อนทำดัชนี (ดัชนีเทอมเซต) เพื่อลดจำนวนโหนดที่ติดต่อเป็นวิธีหนึ่งที่ช่วยแก้ปัญหานี้ อย่างไรก็ตามเมื่อขนาดเทอมเซตสูงยิ่งทำให้ขนาดของดัชนีสูงมากขึ้นแบบทวีคูณ การลดขนาดดัชนีโดยตัดคู่เทอมเซตกับเอกสารที่มีความสัมพันธ์น้อยออกไปต้องแลกกับเอกสารที่ค้นคืนไม่ครบถ้วน สำหรับงานวิจัยนี้ได้ใช้วิธีการจัดหมู่ทำดัชนีโดยจำกัดขนาดเทอมเซตและตัดคู่เทอมเซตกับเอกสารด้วย TF-IDF โดยนำเสนอกลยุทธ์ในการค้นหาเมื่อจำนวนคำค้นหามีมากกว่าขนาดเทอมเซต ซึ่งต้องมีการแตกคีย์เวิร์ดและจับคู่คำค้นหา โดยจะจับคู่คำค้นหาด้วยค่า TFxIDF เปรียบเทียบกับคำค้นหาที่จับคู่แบบสุ่ม จากการทดลองพบว่า การจับคู่คำค้นหาด้วยค่า TFxIDF ทำให้ได้ค่า Recall เพิ่มขึ้น และปริมาณข้อมูลลดลง โดยวัดปริมาณข้อมูล(Bytes)ของรายการเอกสารที่ส่งผ่านเครือข่าย จากการจำลองการค้นหาเอกสารด้วยโปรแกรมจำลองระบบเพียร์ทูเพียร์แบบมีโครงสร้างโดยใช้โปรโตคอล Chord

Thesis	Searching Strategy in Peer-to-Peer System for Multi-Keyword Query
Student	Mr.Tanapon Jitnupong
Student ID.	52660410
Degree	Master of Science
Program	Information Technology
Major	Information System Technology
Year	2014
Thesis Advisor	Assoc. Prof. Dr. Nopporn Chotikakamthorn

ABSTRACT

This thesis describes a strategy for multi-keyword search in peer-to-peer systems. Multi-keyword search in a peer-to-peer system based on DHT with single-term indices causes high amounts of traffic due to the intersection operation of distributed inverted lists. Indexing by a set of keywords (term set) is one possible solution to reduce the communication cost. However, as the number of terms for each set increases, the number of possible term sets increases tremendously. Index pruning partly solves the problem, with the cost of reduced recall rate. This thesis, describe a strategy to search for documents by taking advantage of statistical information which is obtained during indexing for getting more completed document result. The results showed that the recall rate increased and the communication cost is lower than normal search from the pruned index.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ.ดร.นพพร โชติกำธร ที่ได้ให้ความช่วยเหลือ ให้คำแนะนำอันเป็นประโยชน์อย่างยิ่งต่อข้าพเจ้า

ขอกราบขอบพระคุณครูอาจารย์ทุกท่านที่ประสิทธิ์ประศาสตร์วิชาความรู้ให้ข้าพเข้า

ขอขอบคุณพี่น้อง IME-Lab ที่เป็นเพื่อน ให้ความช่วยเหลือและเป็นกำลังใจ ทำให้ข้าพเข้าไม่รู้สึกโดดเดี่ยว

ขอกราบขอบพระคุณป้ากุ๋ย ที่ช่วยเหลือเรื่องภาษาอังกฤษให้ข้าพเจ้า ขอบคุณ พี่จุ๊บจิ๊บ พี่อาร์ท และ หลิง ที่ช่วยเหลือในเรื่องต่างๆและเป็นกำลังใจให้ตลอดมา

คุณความดีอันใดที่ได้จากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้แก่ บิดามารดาที่เคารพรักยิ่งและเป็นกำลังใจที่สำคัญสุดของข้าพเจ้า

ธนพล จิตนุพงศ์

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญตาราง(ต่อ)

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การค้นหาและแบ่งปันข้อมูลผ่านระบบเครือข่ายคอมพิวเตอร์ได้รับความนิยมเพราะมีความสะดวกรวดเร็วและมีจำนวนข้อมูลสารสนเทศมากมายเช่นในเครือข่ายอินเตอร์เน็ต นอกเหนือจากระบบค้นหาแบบเสิร์ชเอนจินทั่วไป ระบบเครือข่ายเพียร์ทูเพียร์เป็นอีกช่องทางหนึ่งในการเชื่อมโยงคอมพิวเตอร์เข้าด้วยกันทำให้สามารถค้นหาและแบ่งปันข้อมูลระหว่างกันได้ ระบบเพียร์ทูเพียร์สามารถรองรับการขยายขนาดของระบบได้ดี นอกจากนี้ระบบเพียร์ทูเพียร์มีข้อดีที่สามารถนำมาประยุกต์ใช้ให้เข้ากับระบบงานหรือวัตถุประสงค์บางอย่างได้โดยเฉพาะ เช่น การแชร์ทรัพยากรในระบบ ไม่ว่าจะเป็นการประมวลผลหรือพื้นที่จัดเก็บข้อมูล เป็นต้น

งานวิจัยในด้านการจัดเก็บและค้นคืนสารสนเทศแบบกระจาย (Distributed Information Retrieval) เป็นส่วนหนึ่งที่คิดนำเอาข้อดีของระบบเครือข่ายเพียร์ทูเพียร์มาใช้ให้เกิดประโยชน์ แต่การประยุกต์ใช้นั้นมีความซับซ้อนกว่าระบบแบบรวมศูนย์ทั่วไป การทำดัชนีหรือการค้นหาข้อมูลบนโหนดที่กระจายกันอยู่ โดยเฉพาะการค้นหาเอกสารแบบหลายคีย์เวิร์ด จะต้องรวบรวมรายการเอกสารจากโหนดต่างๆ อันก่อให้ปริมาณข้อมูลในเครือข่ายสูง การวิจัยและพัฒนาในการค้นหาแบบหลายคีย์เวิร์ดที่ผ่านมาจึงเน้นที่การลดปริมาณข้อมูลที่เกิดขึ้น เช่นการทำดัชนีเทอมเซต แต่ก็ต้องแลกกับพื้นที่ในการจัดเก็บดัชนีที่มีปริมาณมากขึ้น การลดพื้นที่จัดเก็บดัชนีเทอมเซตจะต้องตัดเอกสารบางส่วนไปหรือการคัดออก ทำให้ประสิทธิภาพในการค้นคืนเอกสารที่ลดลง อีกทั้งยังไม่สามารถสร้างเทอมเซตที่มีขนาดใหญ่ได้ เพราะขนาดดัชนีจะเพิ่มเป็นแบบทวีคูณจนเกินกว่าจะยอมรับได้ เมื่อขนาดเทอมเซตถูกจำกัดทำให้การค้นหาที่จำนวนคีย์เวิร์ดมีมากกว่าขนาดเทอมเซตจำเป็นต้องแบ่งคำค้นหาออกเป็นคำค้นย่อยแล้วจึงรวบรวมผลลัพธ์ ในงานวิจัยนี้เน้นการเพิ่มประสิทธิภาพในการค้นคืนเอกสารบนระบบเพียร์ทูเพียร์แบบมีโครงสร้าง ที่ทำดัชนีเทอมเซตแบบมีการคัดออก ในกรณีที่จำนวนคำค้นหามีมากกว่าขนาดเทอมเซต คำค้นหาจะถูกแบ่งย่อยโดยใช้ค่า TFxIDF เป็นตัวช่วย แทนการจับคู่แบบสุ่ม ซึ่งช่วยเพิ่มค่า Recall Precision และลดปริมาณข้อมูลที่เกิดขึ้นในเครือข่ายเพียร์ทูเพียร์แบบ DHT Lookup Table

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

ความมุ่งหมายและวัตถุประสงค์ของการศึกษานี้ เพื่อศึกษาและพัฒนาวิธีการค้นหาแบบหลายคีย์เวิร์ดในระบบเพียร์ทูเพียร์ แบบ DHT Lookup Table ที่มีการจัดทำดัชนีเทอมเซตแบบมีการคัดออก เพื่อปรับปรุงการค้นหาให้มีประสิทธิภาพดีขึ้น ในแง่การค้นคืนเอกสาร และ ลดปริมาณข้อมูลที่เกิดขึ้นในเครือข่ายเพียร์ทูเพียร์

1.3 สมมติฐานของการศึกษา

การค้นหาแบบหลายคีย์เวิร์ดจากดัชนีเทอมเซต ในกรณีที่จำนวนคีย์เวิร์ดในคำค้นมีมากกว่าจำนวนเทอมสูงสุดที่จัดเก็บในดัชนีเทอมเซต จำเป็นต้องมีการแบ่งและจับคู่คำค้นออกเป็นคำค้นย่อย ที่แต่ละคำค้นมีจำนวนคีย์เวิร์ดไม่เกินจำนวนเทอมสูงสุดที่จัดเก็บในดัชนีเทอมเซต สำหรับดัชนีที่มีการตัดทิ้งเทอมเซตบางส่วนเพื่อลดขนาดของตัวดัชนี การแบ่งและจับคู่คำค้นดังกล่าว ส่งผลต่อประสิทธิภาพในการค้นคืนเอกสาร การใช้ค่า TFxIDF มาเป็นปัจจัยในการแบ่งและจับคู่ ช่วยเพิ่มประสิทธิภาพในการค้นคืนได้ดีกว่าการแบ่งและจับคู่คำค้นแบบสุ่ม

1.4 ทฤษฎีและแนวความคิดที่ใช้ในการวิจัย

งานวิจัยนี้อยู่บนพื้นฐานของระบบสืบค้นสารสนเทศแบบกระจาย ที่ใช้เพียร์ทูเพียร์แบบมีโครงสร้าง และใช้ดัชนีเทอมเซตที่มีการคัดออก เมื่อจำนวนคำค้นหามีมากกว่าขนาดเทอมเซต จะแบ่งคีย์เวิร์ดโดยใช้ค่าสถิติที่ใช้ในการจัดลำดับเอกสารเป็นตัวช่วยในการเลือกคีย์เวิร์ด เพื่อเพิ่มประสิทธิภาพในการค้นคืนข้อมูล

1.5 ขอบเขตของการศึกษา

1. มุ่งเน้นการศึกษากับระบบ peer-to-peer แบบมีโครงสร้าง
2. เป็นการศึกษากับเอกสารในรูปแบบของหน้าเว็บ โดยอาศัยชุดข้อมูล INEX2009 Wikipedia collection [10]
3. การจำลองอาศัยโปรแกรม PeerfactSim.Kom [11] โดยเน้นการจำลองในระดับเลเยอร์แอพพลิเคชัน โดยใช้โมเดลการจำลองแบบ DHT Lookup Table (Chord)
4. การศึกษา จะพิจารณาประสิทธิภาพในแง่ Recall กับ Precision และ ปริมาณข้อมูลที่ใช้ในเครือข่าย

1.6 ส่วนประกอบของวิทยานิพนธ์

บทที่1 บทนำ กล่าวถึงความเป็นมาและความสำคัญของปัญหา

บทที่2 ทฤษฎีที่เกี่ยวข้องในงานวิจัย

บทที่3 การค้นหาบนระบบเพียร์ทูเพียร์สำหรับคำค้นหาแบบหลายคีย์เวิร์ด

บทที่4 การทดลอง

บทที่5 บทสรุปและข้อเสนอแนะ

บทที่ 2
ทฤษฎีที่เกี่ยวข้องในงานวิจัย

2.1 เครือข่ายเพียร์ทูเพียร์ (Peer-to-Peer Networks)

ระบบเพียร์ทูเพียร์คือเครือข่ายที่วางซ้อนอยู่บนเครือข่ายอื่น (overlay network) ประกอบไปด้วยโหนดหรือเพียร์จำนวนหนึ่งซึ่งทำหน้าที่เป็นทั้ง server และ client ในเวลาเดียวกัน ระบบเพียร์ทูเพียร์สามารถแบ่งประเภทตามการเชื่อมต่อระหว่างโหนดได้ดังนี้

2.1.1 เครือข่ายเพียร์ทูเพียร์แบบไม่มีโครงสร้าง (Unstructured Peer to Peer Network)

เครือข่ายเพียร์ทูเพียร์แบบไม่มีโครงสร้าง โหนดในระบบจะไม่ถูกขั้นตอนวิธีใดๆมาจัดการการเชื่อมต่อให้เป็นไปตามโครงสร้างหรือเป็นไปตามข้อกำหนด โดยแต่ละเพียร์ หรือโหนด ถูกปล่อยให้เป็นอิสระในการเชื่อมต่อกัน ระบบแบบนี้มีความยืดหยุ่นสูงแต่ก็มีข้อเสียในด้านของประสิทธิภาพในการค้นหาข้อมูล ซึ่งต้องกระจายคำค้นหาไปในเครือข่าย เป็นเหตุให้ปริมาณทราฟฟิกสูงและใช้เวลานานโดยเฉพาะอย่างยิ่งเมื่อระบบมีขนาดใหญ่

2.1.2 เครือข่ายเพียร์ทูเพียร์แบบมีโครงสร้าง (Structured Peer to Peer Network)

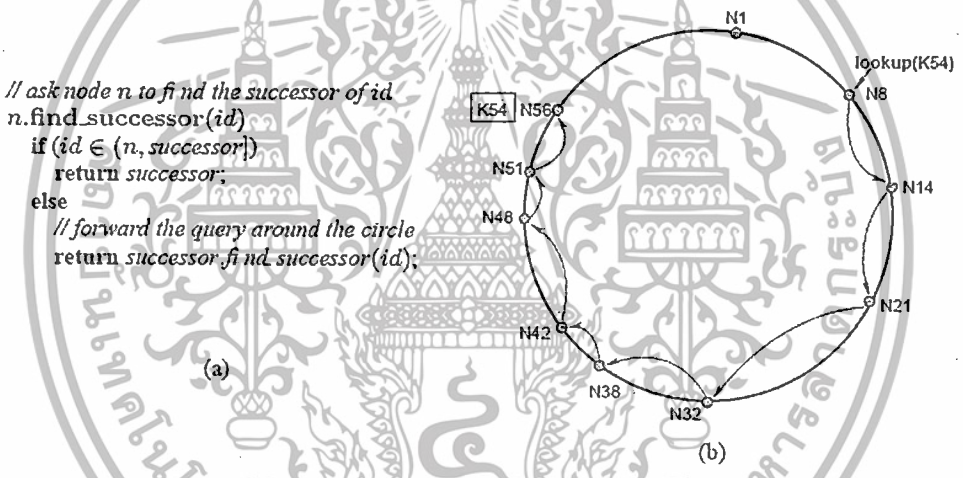
เครือข่ายเพียร์ทูเพียร์แบบมีโครงสร้าง แต่ละโหนดจะถูกจัดการให้ปฏิบัติตามเงื่อนไขข้อกำหนดคือมีโครงสร้างทอโพโลยีที่เคร่งครัด โดยแต่ละโหนดจะมีข้อมูลเกี่ยวกับโหนดอื่นๆจำนวนหนึ่งเพื่อช่วยในการค้นหาเส้นทาง (routing) ด้วย ทำให้การค้นหาข้อมูลในระบบแบบนี้มีประสิทธิภาพ แต่ระบบต้องมีการจัดการเพิ่มขึ้น คือต้องมีการปรับปรุงตารางเส้นทางและจัดการแลกเปลี่ยนข้อมูลเมื่อมีการเข้าออกของโหนดด้วย แต่ทั้งนี้ถ้าจำนวนโหนดในระบบไม่มีการเปลี่ยนแปลงมากระบบแบบนี้จะมีประสิทธิภาพมากกว่าระบบแบบไม่มีโครงสร้างในแง่ของการบริโภคแบนด์วิดธ์ เครือข่ายแบบมีโครงสร้างจะใช้ตารางแฮชเป็นอินเตอร์เฟซและเรียกว่าตารางแฮชแบบกระจาย (Distributed Hash Tables: DHTs)

2.2 ตารางแฮชแบบกระจาย (Distributed Hash Tables)

ตารางแฮชแบบกระจาย (DHTs) คือตารางแฮชที่มีรายการกระจายอยู่ตามโหนดต่างๆ โดยทำหน้าที่บริการการค้นหา (look-up service) เช่นเดียวกับตารางแฮชแบบปกติ คือมีฟังก์ชั่น put สำหรับจัดเก็บคู่ (Key,Value) และ get สำหรับแมปปิ้ง key ไปยัง values ตามลำดับ โดยแต่ละ

โหนดจะแบ่งกันรับผิดชอบบางส่วนของคีย์และเก็บตารางเส้นทาง (Routing table) ของโหนดตัวอื่นๆไว้ด้วย ตัวอย่างของ DHTs เช่น Chord [8]

Chord ใช้ Identifier space เป็นวงกลมแบบ 1 มิติ มีช่วงขนาดตั้งแต่ 0 ถึง 2^m-1 โดยที่ m คือจำนวนบิตที่ใช้สำหรับคีย์และโหนด โหนดที่มีไอดีอยู่ก่อนหน้าและใกล้กับคีย์ k มากที่สุดจะรับผิดชอบ k และเรียกโหนดนั้นๆว่า successor of k แต่ละโหนดจะมีตารางเส้นทางที่เรียกว่า Finger table ซึ่งมีขนาด m แถว มีไว้สำหรับเก็บที่อยู่ของโหนดตัวถัดๆไป โดยแถวแรกจะเป็นโหนดถัดไปที่ใกล้ที่สุดซึ่งจะเป็น successor โดยอัตโนมัติ การค้นหาเส้นทางของ Chord คือ โหนดที่รับคีย์ k มาจะดูตาราง Finger table แถวแรกของตัวเองว่าเป็น successor of k หรือไม่ (มีไอดีอยู่ก่อนหน้า k หรือไม่) ถ้าไม่จะส่งต่อคีย์ไปยังโหนดที่มีไอดีสูงที่สุดในตารางของตนเองแต่ไม่เกินคีย์ k กระบวนการนี้จะทำซ้ำจนกว่าจนกว่าจะเจอโหนดที่เป็น successor of k ดังรูปที่ 2.1 [7]



รูปที่ 2.1 (a) Pseudo code ของ chord สำหรับหา successor of k

(b) แสดงเส้นทางที่ query ใช้ไป โดยเริ่มจากโหนด N8 สำหรับคีย์ k54 โดยใช้ Pseudo code (a)

2.3 การจัดทำดัชนีแบบกระจาย (Distributed Indexing)

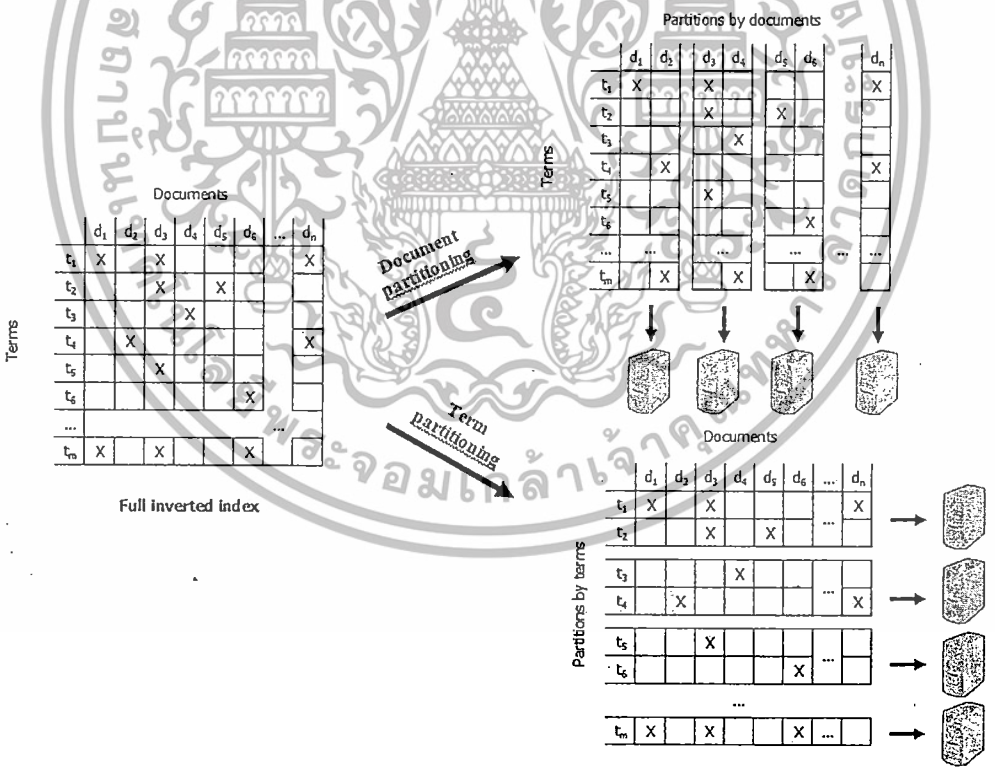
ในระบบค้นหาขนาดใหญ่ที่ไม่สามารถเก็บข้อมูลไว้ในเซิร์ฟเวอร์เดียวได้มีความจำเป็นต้องแบ่งดัชนีกระจายกันเก็บไปยังแต่ละเซิร์ฟเวอร์ ซึ่งสามารถแบ่งเป็นสองวิธีตาม [9] ได้ดังนี้

2.3.1 แบ่งตามเอกสาร (Document Partitioning)

แบ่งเอกสารออกเป็นส่วนย่อยๆ (sub collection) แยกเก็บอิสระต่อกันในแต่ละ server บางครั้งเรียกวิธีนี้ว่า Local Index Partitioning การคิวรี่ของวิธีการนี้ แต่ละ server จะทำขนานไปพร้อมกันแล้วส่งผลลัพธ์กลับมารวมกันอีกครั้ง วิธีการนี้ไม่ซับซ้อนแต่มีโปรเซสซิ่งคอสสูง เพราะต้องกระจายคิวรี่ไปยังทุก server เพื่อทำการค้นหาทั้งหมด

2.3.2 แบ่งตามเทอม (Term Partitioning)

แบ่งเอกสารตามเทอม แต่ละเทอมจะมีรายการเอกสารทั้งหมด (posting list) และจะแยก posting list เก็บคนละ server บางครั้งเรียกวิธีการนี้ว่า Global Index Partitioning การคิวรี่จะติดต่อเฉพาะโหนดที่เก็บ posting list ของเทอมค้นหา ทำให้การค้นหารวดเร็วและไม่ต้องกระจายคิวรี่ไปยังทุกโหนด ระบบP2P ที่ใช้ DHTs นิยมใช้วิธีนี้ เพราะฟังก์ชั่น put/get ประยุกต์ใช้กับ term partitioning ได้ง่าย อย่างไรก็ดี การรวบรวมลัพธ์ของวิธีการนี้ (intersecting posting list) ทำให้เกิดปริมาณทราฟฟิกสูง โดยเฉพาะการค้นหาแบบหลายคีย์เวิร์ด



รูปที่ 2.2 การแบ่งดัชนีตามเทอมและตามเอกสาร

2.4 การให้คะแนนเทอมกับเอกสาร

2.4.1 ความถี่ของเทอม (Term Frequency: TF)

คือการกำหนดน้ำหนัก (weight) ให้แต่ละเทอม t ในเอกสาร d โดยที่น้ำหนักจะขึ้นกับจำนวนครั้งที่เทอมปรากฏในเอกสาร และจะนำมาใช้คำนวณคะแนนระหว่างเทอมค้นหากับเอกสาร การให้น้ำหนักหรือความสำคัญของเทอมส่วนมากจะไม่นับเทอมที่ปรากฏอยู่บ่อยครั้งและไม่มีนัยสำคัญต่อเอกสารหรือที่เรียกว่า stop word ในภาษาอังกฤษเช่นคำว่า a, and, the เป็นต้น ความถี่ของเทอมเขียนแทนด้วย $tf_{t,d}$

2.4.2 ความถี่เอกสารแบบผกผัน (Inverse Document Frequency: IDF)

การนับความถี่ของเทอมอย่างเดียวไม่เพียงพอต่อการประเมินความเกี่ยวข้อง เพราะทุกเทอมถูกพิจารณาว่ามีความสำคัญเท่ากัน แต่เมื่อพิจารณาจากเอกสารที่มีอยู่ทั้งหมด แท้จริงแล้วบางเทอมอาจมีค่าความเฉพาะ (Discriminating power) ต่ำ ในกรณีที่เทอมนั้นๆปรากฏอยู่ทั่วไปแทบทุกเอกสาร ทำให้ไม่อาจสะท้อนความแตกต่างของเอกสารจากเอกสารอื่นได้ วิธีการนี้มีไว้สำหรับลดขนาดค่า TF โดยพิจารณาจากจำนวนเอกสารทั้งหมดที่มีเทอมนั้นๆด้วย ค่า IDF แทนด้วย

$$\text{idf} = \log \frac{N}{df_t} \tag{2.1}$$

เมื่อ N คือจำนวนเอกสารทั้งหมด และ df_t คือจำนวนเอกสารทั้งหมดที่ปรากฏเทอม t

2.4.3 การให้น้ำหนักด้วยความถี่ของเทอมและความถี่เอกสารแบบผกผัน (TF-IDF Weighting)

คือวิธีการให้น้ำหนักแต่ละเทอมในเอกสารโดยใช้ค่า TF และ IDF ประกอบเข้าด้วยกัน ตามวิธีของ G. Salton และ C. Buckley [5] แทนด้วย

$$\text{tf} - \text{idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t \tag{2.2}$$

2.4.4 แบบจำลองเวกเตอร์สเปซ (Vector Space Model)

เป็นวิธีการเปรียบเทียบความคล้าย (similarity) ระหว่างเวกเตอร์สองตัวโดยการวัดมุม cosine โดยแทนแต่ละเทอมในเอกสารให้อยู่ในรูปเวกเตอร์ ขนาดของเวกเตอร์จะเท่ากับจำนวนเทอมที่มีในเอกสาร การเปรียบเทียบจะนำเวกเตอร์มา dot product ถ้าค่าเข้าใกล้หนึ่งแสดงว่าเอกสารมีความสัมพันธ์กัน สมการ cosine similarity ของเอกสาร d_1 และ d_2 แทนด้วย

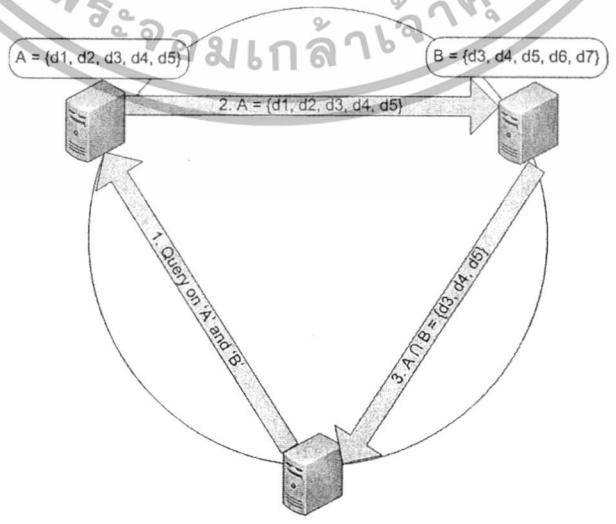
$$sim(d_1, d_2) = \frac{\vec{v}(d_1).\vec{v}(d_2)}{|\vec{v}(d_1)||\vec{v}(d_2)|} \quad (2.3)$$

เราสามารถใช้วิธีการนี้ในการจัดอันดับผลการค้นหาได้ โดยเปรียบเทียบระหว่างคิวรี่กับเอกสาร สมการ cosine similarity ของคิวรี่ q และเอกสาร d แทนด้วย

$$score(q, d) = \frac{\vec{v}(q).\vec{v}(d)}{|\vec{v}(q)||\vec{v}(d)|} \quad (2.4)$$

2.5 การค้นหาแบบหลายคีย์เวิร์ดบนระบบเพียร์ทูเพียร์

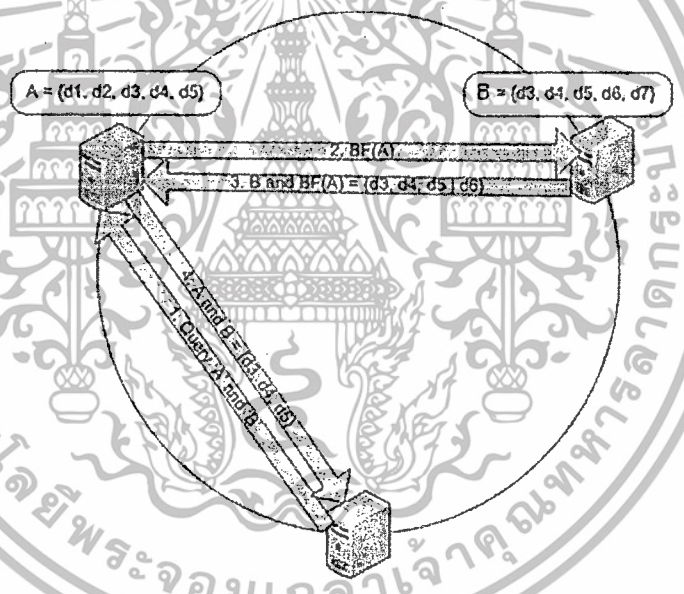
ดัชนีข้อมูลในระบบเพียร์ทูเพียร์แบบมีโครงสร้างจะแบ่งตามเทอมและกระจายกันอยู่ตามโหนดต่างๆ การค้นหาข้อมูล สำหรับคีย์เวิร์ดหรือเทอมเดียวจะนำเทอมเข้าแฮชฟังก์ชั่นของ DHTs เพื่อระบุตำแหน่งโหนดที่เก็บรายการเอกสารของเทอมนั้นๆและจะส่งรายการเอกสารกลับมาโหนดที่ค้นหา สำหรับการค้นหาแบบหลายคีย์เวิร์ด เมื่อระบุโหนดแรกแล้วจะส่งรายการเอกสารไปยัง โหนดที่เก็บเทอมถัดไป เพื่อทำการ intersection รายการเอกสาร และจะส่งต่อไปจนกว่าครบจำนวนเทอมที่ค้นหา สุดท้ายจะส่งผลลัพธ์ของรายการที่กลับมายังโหนดที่เริ่มต้นค้นหา ตามรูปที่ 2.1 คือตัวอย่างขั้นตอนการการรวบรวมรายการเอกสาร โดยใช้คำค้นหาจำนวนสองคีย์เวิร์ด ขั้นตอน 1 query A และ B จะถูกส่งไปที่ node ที่เก็บเทอม A ก่อนแล้วจึงส่งรายการเอกสารของ A ไปให้ node ที่เก็บเทอม B ตามลำดับ จากนั้นจึงทำการ intersection รายการเอกสารของ A และ B และส่งกลับมาให้โหนดที่เริ่มต้น query เป็นขั้นตอนสุดท้าย



รูปที่ 2.3 ขั้นตอนการรวบรวมรายการเอกสาร

ปัญหาของการค้นหาแบบหลายคีย์เวิร์ดบนระบบเพียร์ทูเพียร์แบบมีโครงสร้างคือปริมาณข้อมูลที่สูงจากการรวบรวมรายการเอกสารของแต่ละคีย์เวิร์ดที่กระจายกันอยู่ (distributed intersection) จึงมีผู้ปรับปรุงเทคนิคการค้นหาแบบหลายคีย์เวิร์ดอยู่หลายวิธีดังนี้

1) Reynolds และ Vahdat [3] ใช้ bloom filters [4] แทนรายการเอกสารของเทอมเพื่อลดขนาดรายการเอกสารที่ส่งไปintersection กับรายการเอกสารของเทอมอื่น แต่วิธีการนี้อาจเกิด false positive ทำให้โหนดที่รับต้องส่งรายการเอกสารของตัวเองที่ intersection กับ bloom filter กลับมาโหนดส่งอีกครั้งเพื่อตัดเอกสารที่เกิน ดังรูปที่ 2.2 ขั้นตอนแรก query A และ B จะถูกส่งไปยังโหนดที่เก็บรายการเอกสารของ A ขั้นตอนที่ 2 โหนดที่เก็บ A จะส่ง bloom filter ของ A ให้โหนดที่เก็บรายการเอกสารของ B ขั้นตอนที่ 3 รายการเอกสารของ B จะ intersection กับ BF(A) แล้วส่งกลับมาโหนดที่เก็บ A อีกครั้งเพื่อตัดรายการเอกสารที่เกินออก สุดท้ายโหนดที่เก็บ A จะส่งรายการเอกสารผลลัพธ์กลับมายังโหนดที่เริ่มต้น query



รูปที่ 2.4 ขั้นตอนการรวบรวมรายการเอกสารโดยใช้ Bloom filter [4]

2) Gnawali [1] นำเสนอ The Keyword-Set Search System (KSS) คือการนำคีย์เวิร์ดมาจัดหมู่ (Combination) ก่อนไปทำดัชนียัง DHT เพื่อลดจำนวนโหนดที่ต้องติดต่อระหว่างการค้นหา วิธีการคือกำหนดให้ k เป็นขนาดของเทอมเซตที่ต้องการ KSS จะสร้างดัชนีเทอมเซตสำหรับ k combination จากจำนวนเทอมที่มีทั้งหมดในเอกสาร ดังนั้นจำนวนแถวทั้งหมดของดัชนีจากเอกสารที่มี n เทอมจะได้เท่ากับ $C(n,k) = \frac{n!}{(n-k)!k!}$ ข้อเสียของวิธีการนี้คือทำให้ขนาดของดัชนีใหญ่มากหลายเท่า โดยเฉพาะอย่างยิ่งเมื่อมีการเพิ่มขนาดเทอมเซต

ตารางที่ 2.1 แสดงโครงสร้างดัชนีของ KSS

Key	URL
hash(AB)	http://18.175.6.2/doc1
hash(BC)	http://18.175.6.2/doc1
hash(MN)	http://18.175.6.2/doc2

3) H. Chen [2] นำเสนอ TSS เพื่อลดปัญหาดัชนีเทอมเซตที่ใหญ่เกินไป โดยสังเกตพฤติกรรมของผู้ใช้ Search Engine ซึ่งใช้คำค้นหาจำนวนไม่มากและต้องการเฉพาะผลลัพธ์ที่มีความเกี่ยวข้อง TSS ใช้ Gossip algorithm [6] รวบรวมข้อมูลสถิติของเอกสาร TFxIDF และ Vector Space Model (VSM) คำนวณ similarity โดยตัดเทอมเซตกับเอกสารคู่ที่มี similarity น้อยออกไปเพื่อลดขนาดดัชนี วิธีการของ TSS มีดังนี้

1. นำเทอมในเอกสารทั้งหมดมาเรียงลำดับและทำการจัดหมู่โดยกำหนดขนาดเทอมเซตสูงสุดที่ต้องการ ปกติ TSS กำหนดให้เท่ากับ 3

2. คำนวณและกำหนดค่า TF ให้เป็นน้ำหนักเทอมเซต ค่า IDF ให้เป็นน้ำหนักเอกสาร

3. นำเทอมเซตกับเอกสารเข้าฟังชั่น cosine similarity เพื่อหาความสัมพันธ์เทอมเซทกับเอกสาร

4. เรียงลำดับคะแนนมากไปน้อย โดยที่ถ้าค่าเข้าใกล้หนึ่งแสดงว่าเทอมเซทนั้นๆ มีความสัมพันธ์กับเอกสาร

5. TSS จะเลือกตัดคู่เทอมเซตกับเอกสารที่มีความสัมพันธ์น้อยออกไป เอาเฉพาะ top $\lambda n \log(n)$ term set ที่จะนำไปทำดัชนี โดยที่ n คือ จำนวน keyword ใน เอกสาร λ คือค่าพารามิเตอรไว้ควบคุมปริมาณที่พรุนนึ่ง

TSS ช่วยลดขนาดดัชนีเทอมเซตเป็นอย่างมากแต่ก็ต้องแลกกับการสูญเสียเอกสารที่ไม่ได้ทำดัชนีด้วยเช่นกัน ทำให้วิธีการนี้ต้องแลกกับค่า Recall ที่ลดลง หรือการที่ไม่สามารถค้นคืนเอกสารได้ครบถ้วน

บทที่ 3
กลยุทธ์ในการค้นหาบนระบบเพียร์-ทู-เพียร์สำหรับคำค้นหาแบบหลายคีย์เวิร์ด

3.1 ปัญหาของการค้นหาแบบหลายคีย์เวิร์ดจากดัชนีเทอมเซต

วิธีการจัดทำดัชนีของเอกสารบนระบบเพียร์ทูเพียร์แบบ DHT-based โดยทั่วไป รายการเอกสาร หรือ Posting List จะกระจายอยู่ตามโหนดต่างๆบนเครือข่าย การค้นหาเอกสาร จะนำคีย์เวิร์ดมาผ่านฟังก์ชัน lookup(key) เพื่อระบุโหนดที่เก็บ Posting List ของคำค้นหานั้นๆ คำค้นหาที่มีหลายคีย์เวิร์ด การค้นหาจะต้อง Lookup และส่งรายการเอกสารเอกสารไปทีละโหนดเพื่อทำการอินเตอร์เซกชันจนถึงโหนดที่เก็บคีย์เวิร์ดตัวสุดท้ายแล้วส่งผลลัพธ์กลับมาให้โหนดที่เริ่มต้นค้นหา การค้นหาแบบนี้ ถ้าคำค้นหาประกอบด้วยคีย์เวิร์ดจำนวนมากหรือแบบหลายคีย์เวิร์ด ระบบจะต้องส่งรายการเอกสารไปยังโหนดจำนวนมากตามไปด้วย ซึ่งทำให้ใช้ปริมาณข้อมูลเครือข่ายมาก

3.1.1 ดัชนีเทอมเซต

จากปัญหาการค้นหาของดัชนีเทอมเดียว ที่ทำให้การใช้ปริมาณข้อมูลเครือข่ายมาก จึงมีผู้คิดรวบรวมเทอมไว้ที่โหนดเดียวกันเพื่อลดจำนวนโหนดที่ต้องติดต่อในระหว่างการค้นหาหรือเรียกว่าดัชนีเทอมเซต การสร้างดัชนีเทอมเซต จะต้องมีการคอมบิเนชั่น เทอมแต่ละเทอมในเอกสารตามขนาดที่ต้องการก่อน เช่น เอกสารหนึ่งๆ มีจำนวนเทอมที่แตกต่างกัน 100 เทอม ต้องการเทอมเซตขนาด 2 เทอม จะต้องคอมบิเนชั่น $C_2^{100} = 4950$ จำนวนเทอมเซต ปัญหาของวิธีนี้คือ ยิ่งขนาดของเทอมเซตมาก ขนาดของดัชนีเทอมเซตจะสูงมากแบบทวีคูณ ทำให้ต้องใช้ทรัพยากรในการประมวลผลในการจัดทำดัชนีและใช้พื้นที่จัดเก็บเทอมเซตมากขึ้นตามไปด้วย

3.1.2 การค้นหาจากดัชนีเทอมเซต

เนื่องจากดัชนีเทอมเซตไม่สามารถจัดเก็บทุกเทอมเซตที่เกิดจากการคอมบิเนชันได้ทั้งหมดเพราะทำให้ปริมานดัชนีสูงมาก เมื่อขนาดดัชนีถูกจำกัด อย่างเช่นสองเทอมต่อหนึ่งเทอมเซต คำค้นหาสามคีย์เวิร์ดหรือมากกว่านั้นจะต้องแตกคีย์เวิร์ดให้มีจำนวนลดลง ตัวอย่างเช่น ให้ A, B และ C เป็นคีย์เวิร์ดในคำค้นหา การค้นหาจากดัชนีเทอมเซตสองเทอม สามารถแตกคีย์เวิร์ดสามตัวนี้ได้เป็นสองคู่คีย์เวิร์ด เพื่อแยกค้นหาอิสระจากกันแล้วจึงรวบรวมผลลัพธ์ การแตกและ

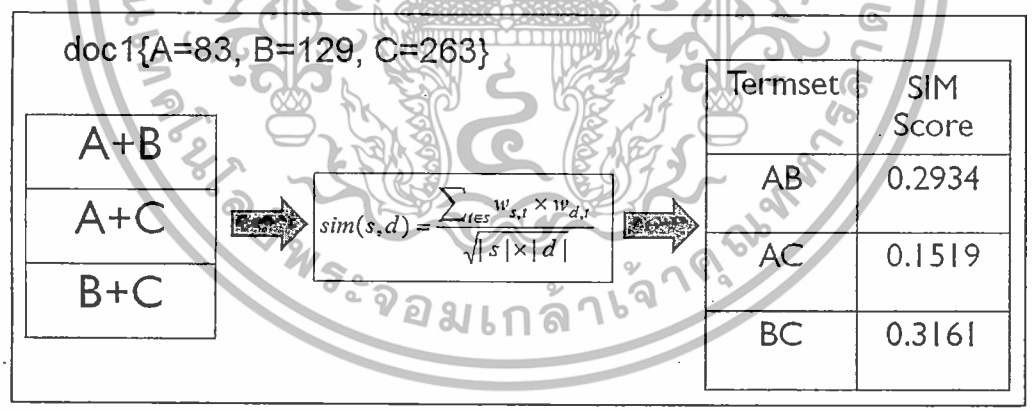
รวบรวมผลลัพธ์สามารถทำได้หลายวิธี เช่น AB ∩ AC, AB ∩ BC และ AC ∩ BC เป็นต้น ดังตัวอย่าง ตามตารางที่ 3.1

ตารางที่ 3.1 การแตกคีย์เวิร์ดและรวบรวมผลลัพธ์

Query term		Combination for Index 2 Term		Choice for Aggregation Operation
A, B, C	→	AB, AC, BC	→	1. AB∩C
				2. AC∩B
				3. BC∩A
				4. AB∩AC
				5. AB∩BC
				6. AC∩BC

3.1.3 การพรุนนิ่งดัชนีเทอมเซต

การสร้างดัชนีเทอมเซตต้องใช้ปริมาณพื้นที่ในการจัดเก็บมากกว่าดัชนีเทอมเดียวหลายเท่า เช่นกำหนด ให้ n เป็นจำนวนเทอมที่แตกต่างกันในหนึ่งเอกสาร กรณีแย่ที่สุดดัชนีเทอมเซตจะมีขนาดถึง 2^n จำนวนเทอมเซต[2] จึงจำเป็นต้องมีการตัดเทอมเซตบางอันออกไปจากดัชนี ซึ่งจากแนวคิดที่ว่าผู้ค้นหาต้องการเฉพาะเอกสารที่เกี่ยวข้อง ดังนั้นคู่เทอมเซตกับเอกสารที่มีความเกี่ยวข้องกับเอกสารน้อยจะถูกคัดออกจากดัชนี โดยใช้สมการหาค่าความคล้าย (cosine similarity) ระหว่างเทอมเซต s กับเอกสาร d โดยใช้ TFxIDF ในการให้น้ำหนัก

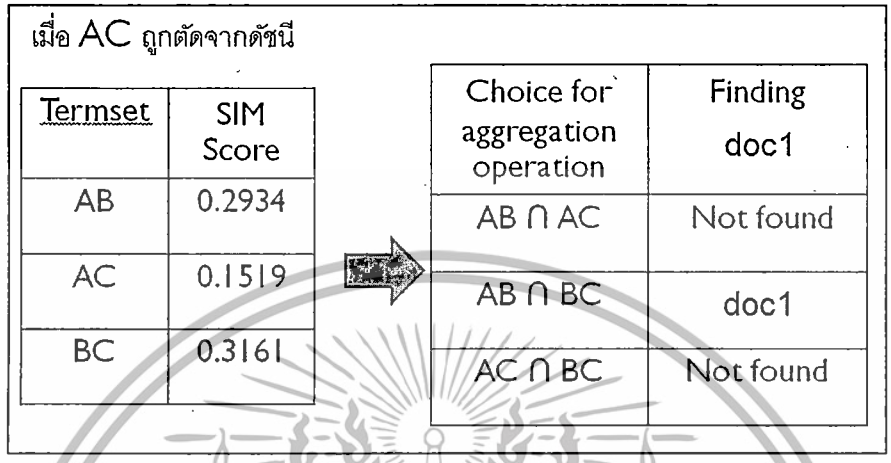


รูปที่3.1 แสดงขั้นตอนการหาค่าความคล้ายของเทอมเซตขนาดสองเทอมต่อเอกสาร doc1 จากรูปเทอมเซต AC มีคะแนนน้อยที่สุดและจะถูกคัดออกจากดัชนี

3.1.4 ปัญหาจากดัชนีเทอมเซตที่มีการคัดออกในกระบวนการพรุนนิ่ง

กรณีที่ดัชนีไม่ถูกคัดออก ดังตารางที่ 3.1 ทุกวิธีจะให้ผลลัพธ์เดียวกัน อย่างไรก็ตาม ถ้าดัชนีมีการคัดออกวิธีนี้ไม่สามารถใช้ได้ ตัวอย่างเช่น ถ้า A เป็นคีย์เวิร์ดที่ไม่ดี (มีคะแนน TFxIDF ต่ำ) เมื่อเลือกกระทำ AB ∩ AC เอกสารที่ประกอบด้วยคีย์เวิร์ดที่ดีกว่า อย่าง B และ C (มีคะแนน

TFxIDF สูงกว่า A) อาจไม่ปรากฏในผลลัพธ์ที่ทำการรวบรวม ในกรณีเช่นนี้ เอกสารที่มีความเกี่ยวข้องถูกคาดหวังว่าจะปรากฏในผลการค้นหา ไม่ว่าจะเลือกเป็น $AB \cap BC$ หรือ $AC \cap BC$ ดังนั้นการเลือกคีย์เวิร์ดที่เหมาะสมเพื่อจับคู่จึงเป็นทางออกที่จะแก้ไขปัญหานี้ เพื่อให้ได้ผลลัพธ์กลับคืนครบถ้วนที่สุด



รูปที่ 3.2 แสดงปัญหาจากการค้นหาจากดัชนีเทอมเซตที่มีการคัดออก

3.2 กลยุทธ์การค้นหาจากดัชนีเทอมเซตที่มีการคัดออก

กลยุทธ์ที่นำเสนอนี้คือการเลือกคีย์เวิร์ดที่เป็นตัวแทนของเอกสาร เพื่อนำไปค้นหายังดัชนีเทอมเซต โดยที่ดัชนีเทอมเซตจะมีการคัดออกตามสมการที่ (3.3) คีย์เวิร์ดที่เป็นตัวแทนของเอกสารคือคีย์เวิร์ดที่พบได้น้อยในเอกสารอื่นๆ หรือกล่าวได้อีกทางหนึ่งคือคีย์เวิร์ดที่มีคะแนนสูงโดยการใช้ TFxIDF ในการให้น้ำหนัก ดังนั้นเทอมเซตที่มีคีย์เวิร์ดเหล่านี้อยู่ มีโอกาสที่จะไม่ถูกคัดออก การจับคู่คีย์เวิร์ดเหล่านี้ในการค้นหาจะทำให้ Recall rate สูงขึ้น ผู้วิจัยได้ใช้ค่า IDF ของคีย์เวิร์ดตัวที่ค่ามากที่สุดเป็นตัวแทนของเอกสาร เพราะเป็นค่า global statistic ซึ่งโหนดที่ค้นหาทราบและค่า IDF บ่งบอกถึงความเฉพาะของเอกสาร ถ้า IDF มีค่ามากแสดงว่าเอกสารที่มีเทอมนั้นมีความเกี่ยวข้องกับคำค้นหาและมีโอกาสสูงที่จะไม่ถูกคัดออกจากดัชนี

3.2.1 ขั้นตอนการแตกคีย์เวิร์ด

เมื่อโหนดได้รับคำค้นหาที่มีจำนวนเท่ากับสามหรือมากกว่า กระบวนการแตกคีย์เวิร์ดจะเริ่มทำงาน กลยุทธ์ในการค้นหาแบบหลายคีย์เวิร์ดมีขั้นตอนดังนี้

1) จัดแบ่งกลุ่มคีย์เวิร์ดทุกความเป็นไปได้ดังเงื่อนไขต่อไปนี้
   1.1) เรียงจัดหมู่โดยกำหนดขนาดไม่เกินขนาดเทอมเซต (lmax)
   1.2) เรียงจับคู่คิวรีให้มีจำนวนน้อยที่สุด

Search {A, B, C}

No.	1	2
1	AB	C
2	AC	B
3	BC	A
4	AB	AC
5	AB	BC
6	AC	BC

รูปที่ 3.3 แสดงจัดแบ่งกลุ่มคีย์เวิร์ด A, B, C ตามเงื่อนไข เมื่อดัชนีเทอมเซตมีขนาด (lmax) เท่ากับสอง

2) ใช้ค่า IDF ให้น้ำหนักแต่ละคู่คีย์เวิร์ด แถวเดียวกันคอลัมภ์ที่ 1 และ 2 คือหนึ่งคู่

1	2	IDF 1	IDF 2
AB	C	3.92	5.61
AC	B	4.71	4.06
BC	A	4.83	3.79
AB	AC	3.92	4.71
AB	BC	3.92	4.83
AC	BC	4.71	4.83

รูปที่ 3.4 แสดงการใช้ค่า IDF ให้น้ำหนักในแต่ละคู่คีย์เวิร์ด

3) ในแต่ละคู่คีย์เวิร์ดเลือกค่าที่น้อยกว่ามาเป็นตัวแทน

IDF 1	IDF 2	min(IDF1,IDF2)
3.92	5.61	3.92
4.71	4.06	4.06
4.83	3.79	3.79
3.92	4.71	3.92
3.92	4.83	3.92
4.71	4.83	4.71

รูปที่ 3.5 แสดงการเลือกตัวแทนจากค่าที่น้อยกว่า

4) จากตัวแทนค่าน้อยเลือกค่ามากที่สุด

No.	1	2	IDF1	IDF2	min(IDF1,IDF2)
1	AB	C	3.92	5.61	3.92
2	AC	B	4.71	4.06	4.06
3	BC	A	4.83	3.79	3.79
4	AB	AC	3.92	4.71	3.92
5	AB	BC	3.92	4.83	3.92
6	AC	BC	4.71	4.83	4.71

รูปที่ 3.6 แสดงการเลือกคีย์เวิร์ดที่มีค่ามากที่สุดจากตัวแทน

จากรูปที่ 3.6 กำหนดให้แถวที่ 6 คือ CA, CB เป็นตัวแทนที่จะใช้ค้นหาจากดัชนีเทอมเซต ซึ่งจากขั้นตอนที่กล่าวมา การเลือกคู่ใดๆสามารถเขียนให้อยู่ในรูปสมการ

$$\arg\max_{i \in S} (\min(IDF1_i, IDF2_i)) \tag{3.1}$$

ตารางที่ 3.2 แสดงจำนวนโอเปอเรชันที่ใช้ในขั้นตอนการเลือกคีย์เวิร์ด

ขั้นตอน	จำนวนโอเปอร์เรชัน (ครั้ง)	
	จำนวนคำค้นหา = 3	จำนวนคำค้นหา = 4
1) จัดแบ่งกลุ่มคีย์เวิร์ดทุกความเป็นไปได้	8	16
2) ใช้ค่า IDF ให้น้ำหนักแต่ละคู่คีย์เวิร์ด	3	4
3) ในแต่ละคู่คีย์เวิร์ดเลือกค่าที่น้อยกว่ามาเป็นตัวแทน	16	32
4) เลือกคีย์เวิร์ดที่มีค่ามากที่สุดจากตัวแทน	12	20

3.2.2 ขั้นตอนการรวบรวมและจัดอันดับลำดับผลการค้นหา

เมื่อขั้นตอนการแตกคีย์เวิร์ดเสร็จสิ้นแล้ว จะทำการค้นหาและรวบรวมข้อมูลดังต่อไปนี้

1) สำหรับแต่ละคู่คีย์เวิร์ด จะนำไปเข้าฟังก์ชั่นแฮชเพื่อสร้างเป็น key และนำ key ไปดึงรายการเอกสารจากดัชนีเทอมเซตตามโหนดที่เก็บรายการเอกสารทีละคู่ หลังจากนั้นเอกสารจะถูกรวบรวม โดยการ intersection
2) สำหรับรายการเอกสารที่เป็นผลลัพธ์จากขั้นตอนก่อนหน้า นำมาหา similarity ระหว่างคำค้นหากับเอกสาร ตามสมการ (3.3) เพื่อจัดลำดับผลการค้นหา โดยเปลี่ยนจากเทอมเซตเป็นคำค้นหาแทน

3.3 การสร้างดัชนีเทอมเซต

ดัชนีเทอมเซตคือดัชนีที่นำเทอมมาจัดหมู่ก่อนทำดัชนี เพื่อลดการติดต่อของโหนดระหว่างการค้นหา ซึ่งมีขั้นตอนในการสร้างดังนี้[2]

1) นำเอกสารมาเก็บคะแนนความถี่ของเทอมที่ปรากฏและจัดทำดัชนีเก็บใน local index โดยคำนวณค่า TF ตามสมการ

$$TF_{t,d} = 1 + \log(f_{t,d}) \tag{3.2}$$

โดยที่ $f_{t,d}$ คือจำนวนความถี่ของเทอม t ที่ปรากฏในเอกสาร d

2) นำค่าสถิติ global ซึ่งได้มาจาก gossip algorithm มาคำนวณค่า IDF ตามสมการ

$$IDF_t = \log(1 + \frac{N}{f_t}) \tag{3.3}$$

โดยที่ f_t คือจำนวนเอกสารทั้งหมดที่มีเทอม t และ N คือจำนวนเอกสารทั้งหมดในระบบ

3) ในแต่ละเอกสาร เทอมจะถูกเรียงตามตัวอักษรและคำนวณจัดหมู่เทอมเซตครั้งละสอง ($C(n,2)$) โดยที่ n คือจำนวนเทอมที่มีทั้งหมดในเอกสาร จากนั้นจะคำนวณหาค่าความสัมพันธ์ระหว่างเทอมเซตกับเอกสาร เพื่อตัดขนาดของดัชนีตามวิธีของ chen ดังนี้ แต่ละเทอมเซต s จะกำหนดให้ $w_{s,t} = IDF_t$ เป็นน้ำหนักของเทอมในสมาชิก และกำหนดให้ $w_{d,t} = TF_{t,d}$ เป็นน้ำหนักของเอกสาร โดยที่ $w_{s,t}$ คือค่าน้ำหนักของเทอมเซต และ $w_{d,t}$ คือค่าน้ำหนักของเอกสาร d ตามลำดับ จากนั้นหาค่าความสัมพันธ์ (similarity) ระหว่างเทอมเซตกับเอกสาร ตามสมการ

$$sim(s,d) = \frac{\sum_{t \in s} w_{s,t} \times w_{d,t}}{\sqrt{|s| \times |d|}} \tag{3.4}$$

โดยที่ |s|และ|d| คือจำนวนเทอมใน s และ d ตามลำดับ

4) เมื่อได้คะแนนความสัมพันธ์ระหว่างเทอมเซตกับเอกสารแล้วให้นำมาเรียงลำดับจากมากไปน้อย แล้วเลือกเอาเฉพาะ top $\lambda nlog(n)$ เทอมเซต เพื่อจัดเก็บไปยังดัชนีโกลบอล (DHTs) โดยที่ n คือจำนวนเทอมที่มีทั้งหมดในเอกสาร

Algorithm: Index generating and pruning

input: document vector
output: pruned term-set vector

for each document *d* in collection
$n \leftarrow$ number of terms in *d*
$s \leftarrow$ combination of terms in *d (C(n,2))*
for each term-set *s*
$w_{d,t} \leftarrow \{TF_{t1}, TF_{t2}\}$
$w_{s,t} \leftarrow \{IDF_{t1}, IDF_{t2}\}$
$sim_{s,d} \leftarrow sim(w_{d,t}, w_{s,t})$
end for
rank{ $sim_{s,d}$ } in a descending
truncate { $sim_{s,d}$ } at position that less than top
$\lambda n log(n)$
end for

รูปที่ 3.7 แสดงอัลกอริทึมที่ใช้ในการสร้างดัชนีเทอมเซต

3.4 การลดปริมาณข้อมูลที่ใช้ในเครือข่าย

การลดปริมาณข้อมูลที่ใช้ในเครือข่ายสำหรับการค้นหาแบบหลายคีย์เวิร์ดแบ่งเป็นสองส่วนดังนี้

3.4.1 การจัดลำดับคำค้นหา

ในการค้นหาแบบหลายคีย์เวิร์ดบนระบบเพียร์ทูเพียร์แบบ DHT-based จะต้องมีการอินเตอร์เซกชันรายการเอกสารตามโหนดต่างๆไปทีละคำค้นหา เนื่องจากผลลัพธ์จากการอินเตอร์เซกชันจะน้อยลงไปเรื่อยๆและผลลัพธ์สุดท้ายรายการเอกสารจะมีจำนวนไม่เกินรายการเอกสารของคีย์เวิร์ดตัวที่น้อยที่สุด ดังนั้นคำค้นหาเดียวกันไม่ว่าจะเรียงลำดับคีย์เวิร์ดแตกต่างกันอย่างไรจะไม่มีผลต่อผลลัพธ์หรือรายการเอกสารที่ได้จากการอินเตอร์เซกชันแต่ปริมาณข้อมูลในเครือข่ายจะแตกต่าง ดังนั้นถ้าเรียงลำดับโดยเริ่มต้นค้นหาด้วยคีย์เวิร์ดที่มีรายการเอกสารน้อยก่อนจะทำให้ลดปริมาณข้อมูลในเครือข่ายได้ [14]

3.4.2 การเลือกจับคู่คำค้นหา

การเลือกจับคู่คำค้นหาจากดัชนีเทอมเซต จากกลยุทธ์ในการค้นหาที่อธิบายในหัวข้อ 3.2 นอกเหนือจากโอกาสที่จะได้เอกสารไม่ถูกคัดออกแล้ว การเลือกจับคู่จากค่า IDF มีผลทำให้ได้เอกสารที่มีความเฉพาะสูง ทำให้การค้นหาแบบหลายคีย์เวิร์ดจะเริ่มตั้งต้นด้วยเทอมเซตที่มีปริมาณรายการเอกสารน้อย มีผลต่อลำดับการอินเตอร์เซกชันที่จะทำให้ปริมาณข้อมูลลดลงได้

บทที่ 4

การทดลอง

4.1 การวัดประสิทธิภาพการค้นหา

4.1.1 ข้อมูลที่ใช้ทดลอง

ข้อมูลที่ใช้ทดลองคือ INEX 2009 Wikipedia collection [10] จากการรวบรวมของ Max Planck Institute of Informatik ประเทศเยอรมัน ข้อมูลในแต่ละเอกสารจัดเก็บอยู่ในรูปแบบ XML มีขนาดทั้งสิ้น 50 .7 GB จำนวน 2,666,190 บทความ ข้อมูลชุดที่ 1 ใช้เอกสารจำนวน 24,500 ทำดัชนี ข้อมูลชุดที่ 2 ใช้เอกสารจำนวน 148,000 เอกสาร

4.1.2 การสร้างคิวรี่

Query ที่ใช้ทดสอบมาจาก https://inex.mmci.uni-saarland.de/Inex 2009 topics จำนวน 115 topics เอกสารอยู่ในรูปแบบ XML เลือกมาโดยการสกัดคีย์เวิร์ดที่อยู่ในแต่ละ topics โดยตัดคีย์เวิร์ดที่เป็น stop word ออกไป ซึ่งจะได้ข้อมูลจำนวนคีย์เวิร์ดที่แตกต่างกัน 1,459 คีย์เวิร์ด จำนวนคีย์เวิร์ดเฉลี่ยต่อ topics เท่ากับ 23.06 คีย์เวิร์ด

การสร้างคิวรีสามคีย์เวิร์ด ในแต่ละ topic จะทำการเรียงจัดหมู่ (Combination) คีย์เวิร์ดทีละสาม และทำการสุ่มเลือกจากแต่ละ topic เป็นคิวรีชุดที่ 1 จำนวน 98 คิวรี และชุดคิวรีที่2 จำนวน 136 คิวรี เพื่อนำมาใช้ทดลอง โดยคิวรีนี้จะนำไปค้นหาจากดัชนีแบบ Centralized TFxIDF เพื่อให้ได้เอกสารที่ครบถ้วนไม่มีการคัดออก ซึ่งจะนำมาใช้เป็นข้อมูล Ground Truth เพื่อทดสอบเปรียบเทียบกับดัชนีแบบเทอมเซตที่ได้สร้างขึ้น คิวรีชุดที่ 1 นำไปค้นหาหาจากดัชนีแบบ Centralized TFxIDF ได้จำนวนเอกสารที่แตกต่างกัน 24,605 เอกสาร จำนวนเอกสารเฉลี่ยที่ค้นคืนได้ต่อหนึ่งคิวรีเท่ากับ 514.52 เอกสาร

สำหรับการคิวรีจากดัชนีเทอมเซต เนื่องจากดัชนีเทอมเซตที่ใช้ในงานวิจัยนี้ กำหนดให้มีขนาดของเทอมเซตเท่ากับ 2 ในการค้นหาคิวรีแบบ 3 คีย์เวิร์ด จึงต้องมีการเรียงแยกคีย์เวิร์ดเป็นสองคู่แล้วจึงทำการอินเตอร์เซกชันรวบรวมผลลัพธ์ ในการทดลองนี้ได้แบ่งคิวรีเป็นสองกลุ่มดังนี้คือ

1.) แบบ 2∩1 หมายถึงคีย์เวิร์ดสองคีย์เวิร์ดจับคู่กัน (เทอมเซต) อินเตอร์เซกชันกับคีย์เวิร์ดเดียว (เทอมเดียว)

2.) แบบ 2∩2 หมายถึงคีย์เวิร์ดสองคีย์เวิร์ดจับคู่กัน(เทอมเซตคู่แรก) อินเตอร์เซกชัน คีย์เวิร์ดสองคีย์เวิร์ดจับคู่กัน (เทอมเซตคู่ที่สอง)

4.1.3 ค่า Recall และ Precision

หาค่า Recall และ Precision โดยทำการค้นหาเปรียบเทียบคิวรีที่ไม่มีการเลือกจับคู่และแบบวิธีที่เลือกจับคู่โดยเลือกเทอมที่มีค่า IDF มากสุด คำนวณค่า Recall และ Precision โดยใช้ผลลัพธ์ทั้งหมดที่ได้จาก ดัชนีเทอมเดียวเป็น Ground truth เนื่องจากดัชนีเทอมเดียวไม่มีการ pruning ทำให้มีเอกสารครบทั้งหมด ทดลองเปรียบเทียบค่า Recall และ Precision ตามสมการที่ 4.1

$$\text{Recall} = \frac{\text{ผลลัพธ์จากดัชนีเทอมเซต} \cap \text{ผลลัพธ์จากดัชนีเทอมเดียว } 30 \text{ อันดับสูงสุด}}{\text{ผลลัพธ์จากดัชนีเทอมเดียว } 30 \text{ อันดับสูงสุด}}$$

$$\text{Precision} = \frac{\text{ผลลัพธ์จากดัชนีเทอมเซต} \cap \text{ผลลัพธ์จากดัชนีเทอมเดียว } 10 \text{ อันดับสูงสุด}}{\text{ผลลัพธ์จากดัชนีเทอมเซต}} \qquad (4.1)$$

4.1.4 ความแตกต่างของสูตรที่ใช้ทำดัชนี

จากสมการที่ 3.4 ในบทที่ 3 ในการจัดลำดับเทอมเซตที่สัมพันธ์กับเอกสารสำหรับใช้ในการพรุนนิ่งดัชนีเทอมเซต ส่วนที่ต้อง Normalized ด้วยจำนวนของเทอมเซต โดยการหารด้วย $\sqrt{|s|}$ ผู้วิจัยได้ทดลองตัด $\sqrt{}$ ออกและหารด้วย $|s|$ แทน เพื่อทดสอบประสิทธิภาพในการค้นคืน โดยกำหนดให้ดัชนีแบบที่ 1 มีการ Normalized จำนวนเทอมเซตโดยการหารด้วย $\sqrt{|s|}$ และดัชนีแบบที่ 2 กำหนดให้ Normalized จำนวนเทอมเซตโดยการหารด้วย $|s|$ เพื่อทดสอบความแตกต่างของผลลัพธ์เมื่อมีการค้นหา ทดลองจัดทำดัชนี โดยกำหนดให้ขนาดเทอมเซต $|s| = 2$ เปรียบเทียบทั้งสองแบบที่ขนาดดัชนีเท่ากับ 0.5 และ 1.0 การค้นหาจะใช้วิธีการเลือกจับคู่ด้วย IDF เปรียบเทียบประสิทธิภาพในการค้นหา

ตารางที่ 4.1 แสดงเปรียบเทียบผลลัพธ์จากการค้นแบบ $2 \cap 1$ ดัชนีแบบที่1 และแบบที่ 2

ขนาดดัชนี (λ)	ดัชนีแบบที่ 1 (หารด้วย $\sqrt{\lvert s \rvert}$)		ดัชนีแบบที่ 2 (หารด้วย $\lvert s \rvert$)	
	Recall	Precision	Recall	Precision
0.5	0.0082	0.0194	0.1667	0.2490
1.0	0.0337	0.0694	0.2918	0.3888

ตารางที่ 4.2 แสดงเปรียบเทียบผลลัพธ์จากการค้นแบบ $2 \cap 2$ ดัชนีแบบที่1 และแบบที่ 2

ขนาดดัชนี (λ)	ดัชนีแบบที่ 1 (หารด้วย $\sqrt{\lvert s \rvert}$)		ดัชนีแบบที่ 2 (หารด้วย $\lvert s \rvert$)	
	Recall	Precision	Recall	Precision
0.5	0.2741	0.3745	0.2684	0.3663
1.0	0.4184	0.5143	0.4133	0.5102

จากการเปรียบเทียบดัชนีสองแบบ ตารางที่ 4.1 ดัชนีแบบที่ 2 มี Recall โดยเฉลี่ยสูงกว่าแบบที่ 1 0.20 และ Precision สูงกว่า 0.23ในขณะที่ตาราง 4.2 ดัชนีแบบที่ 2 มีค่า Recall เฉลี่ยต่ำกว่าเพียง 0.054 และ 0.062 เนื่องจากดัชนีแบบที่ 1 มีการตัดเทอมเดียวออกไปจำนวนมากทำให้ผลลัพธ์แบบ $2 \cap 1$ มีค่าลดลงอย่างยิ่ง ในขณะเดียวกัน Recall ในแบบ $2 \cap 2$ ก็ไม่ได้เพิ่มขึ้นตามสัดส่วนที่สูญเสียไป ในงานวิจัยนี้ได้จัดทำดัชนีในแบบที่ 2 คือ Normalized ด้วย $\lvert s \rvert$

4.1.5 ผลการทดลองค้นหาแบบ 3 คีย์เวิร์ด

ทำการทดลองค้นหาแบบ 3 คีย์เวิร์ดโดยใช้จำนวนเอกสารและคิวรีในชุดที่สอง จำนวนคิว เปรียบเทียบวิธีเลือกจับคู่ด้วย IDF ($2 \cap 2$ IDF) กับวิธีแบบสุ่มจับคู่ ($2 \cap 2$ Random) และเปรียบเทียบกับผลลัพธ์ที่ดีที่สุด ($2 \cap 2$ MAX) และแย่ที่สุด ($2 \cap 2$ MIN) ทดลองที่ขนาดดัชนีเท่ากับ 0.1, 0.5, 1.0 และ 1.5 ตามลำดับ

ตารางที่ 4.3 ค่า Recall ผลการค้นหาแบบ 3 คีย์เวิร์ด รูปแบบ 2∩2

ขนาดดัชนี (λ)	Recall			
	2∩2 IDF	2∩2 Random	2∩2 MIN	2∩2MAX
0.1	0.0593	0.0348	0.0157	0.0657
0.5	0.2272	0.1314	0.0659	0.2463
1.0	0.4130	0.2414	0.1221	0.4424
1.5	0.5123	0.3147	0.1740	0.5436

ตารางที่ 4.4 ค่า Precision ของผลการค้นหาแบบ 3 คีย์เวิร์ด รูปแบบ 2∩2

ขนาดดัชนี (λ)	Precision@top10			
	2∩2 IDF	2∩2 Random	2∩2 MIN	2∩2MAX
0.1	0.0993	0.0596	0.0294	0.1110
0.5	0.3426	0.1882	0.1066	0.3691
1.0	0.5368	0.3243	0.1757	0.5750
1.5	0.6184	0.3971	0.2338	0.6537

ตารางที่ 4.5 แสดงประสิทธิภาพที่เพิ่มขึ้น ของผลการค้นหา 2∩2 จับคู่ด้วย IDF เปรียบเทียบกับจับคู่แบบสุ่ม

ขนาดดัชนี (λ)	Recall เพิ่มขึ้น	Precision เพิ่มขึ้น
0.1	70.40%	66.61%
0.5	72.91%	82.04%
1.0	71.09%	65.53%
1.5	62.79%	55.73%

จากตารางที่ 4.4 และ 4.5 แสดงผลการค้นหาแบบ 3 คีย์เวิร์ดของวิธีเลือกจับคู่ด้วย IDF และวิธีเลือกจับคู่แบบสุ่ม เปรียบเทียบกับผลที่ดีและแย่ที่สุด ตารางที่ 4.6 แสดงความแตกต่างร้อยละของประสิทธิภาพที่เพิ่มขึ้นของวิธีเลือกจับคู่ด้วย IDF เปรียบเทียบกับวิธีเลือกจับคู่แบบสุ่ม โดยคำนวน

ดังนี้ (ค่าที่ได้จากวิธี IDF – ค่าที่ได้จากวิธีแบบสุ่ม) / ค่าที่ได้จากวิธีแบบสุ่ม จากตารางที่ 4.6 แสดงให้เห็นว่า วิธีเลือกจับคู่ด้วย IDF มีค่า Recall เพิ่มขึ้นเฉลี่ย 69.30% และ Precision เพิ่มขึ้นเฉลี่ย 67.48% โดยมีแนวโน้มลดลงเมื่อขนาดดัชนีสูงขึ้น

4.1.6 เปรียบเทียบคิวรีที่เป็นแบบ $2 \cap 1$ และแบบ $2 \cap 2$

เปรียบเทียบจากจำนวนคิวรีจำนวน 136 คิวรี ในการค้นหาแบบสามคีย์เวิร์ด ด้วยวิธีเลือกจับคู่ แบบ $2 \cap 1$ และ $2 \cap 2$ เพื่อทราบจำนวนของรูปแบบคิวรีที่ได้รับการคัดเลือก

ตารางที่ 4.6 เปรียบเทียบรูปแบบคิวรีที่ได้รับการคัดเลือกด้วย IDF

ขนาดดัชนี	จำนวนคิวรีที่เลือก			
	$2 \cap 1$		$2 \cap 2$	
	จำนวน	คิดเป็น %	จำนวน	คิดเป็น %
1.0	6	4.41%	130	95.58%

ตารางที่ 4.7 เปรียบเทียบค่าแบบที่เลือกการแตกที่ดีที่สุดกับการแตกแบบ $2 \cap 2$ โดยวิธีเลือกจับคู่ด้วยค่า IDF

ขนาดดัชนี	รูปแบบการเลือกคีย์เวิร์ด			
	$2 \cap 1$ และ $2 \cap 2$		$2 \cap 2$	
	Recall	Precision	Recall	Precision
1.0	0.4172	0.5419	0.4130	0.5368

จากตารางที่ 4.3 สังเกตุได้ว่าคิวรีกว่า 95.58% มีการแตกคีย์เวิร์ดที่เป็นแบบ $2 \cap 2$ เนื่องจาก $2 \cap 1$ ส่วนที่เป็น 1 คือเทอมเดียวมีอยู่ในดัชนีเป็นจำนวนน้อย แม้จะทำดัชนีโดยเพิ่มจำนวนเทอมเดียวดังหัวข้อที่ 4.1.4 ก็ตาม โดยผลจากการค้นหาดังตารางที่ 4.4 วิธีเลือกจับคู่โดยโดยรูปแบบที่ดีที่สุดให้ Recall เพิ่มขึ้นกว่าแบบ $2 \cap 2$ 1.02% และ Precision เพิ่มขึ้น 0.95%

4.2 การจำลองเครือข่ายและการวัดปริมาณข้อมูล

4.2.1 การจำลองการค้นหาดัชนีเทอมเซต

กำหนดให้โหนดในเครือข่ายมีภาระในการรับส่งรายการเอกสารหรือ Posting List โดยกำหนดให้แต่ละ Posting ของ ดัชนีมีโครงสร้างดังนี้ [2]

Key(s)={docID,{f1,f2},|d|}

Posting ของ ดัชนีจะประกอบด้วย docID คือค่า checksumได้จาก URL ของเอกสาร[12] f1,f2 คือจำนวนความถี่ของคีย์เวิร์ด k1,k2 ที่ปรากฏในเอกสาร |d| คือจำนวนเทอมทั้งหมดในเอกสาร d กำหนดให้แต่ละ posting มีขนาด 224 bit แบ่งเป็น docID ขนาด 128 bit และ f1, f2, |d| อย่างละ 32 bit ตัวอย่างเช่น เทอมเซตอันหนึ่งมีเอกสาร 10 เอกสาร posting list จะมีขนาดเท่ากับ 224x10 = 2,240 bit หรือ 280 bytes

4.2.2 ปริมาณข้อมูลของการค้นหาจากดัชนีเทอมเดียวแบบที่เรียงและไม่เรียงลำดับคำค้นหา เปรียบเทียบกับดัชนีแบบเทอมเซต

ทำการเปรียบเทียบการค้นหาจากดัชนีเทอมเดียวที่เรียงและไม่เรียงลำดับคีย์เวิร์ดเปรียบเทียบกับดัชนีแบบเทอมเซต คีย์เวิร์ดที่เรียงลำดับคำค้นหาจะเรียงลำดับคำค้นหา ตามจำนวนเอกสาร เนื่องด้วยปริมาณเอกสารผลลัพธ์จากการอินเตอร์เซกชันจะถูกจำกัดด้วยจำนวนเอกสารของตัวที่น้อยกว่าเสมอ ดังนั้นถ้าเริ่มต้นค้นหาด้วยจำนวนเอกสารที่น้อยก่อนจะมีผลทำให้ปริมาณข้อมูลลดลงได้ จึงนำมาเปรียบเทียบกับการค้นหาด้วยดัชนีเทอมเซต เพื่อเปรียบเทียบปริมาณข้อมูลของทั้งสองแบบ และเนื่องจากดัชนีเทอมเซตมีจำนวนเอกสารผลลัพธ์กลับคืนมาน้อยกว่าสาเหตุจากการที่ดัชนีมีการคัดออก ดังนั้นเพื่อให้การเปรียบเทียบปริมาณข้อมูลมีความเท่าเทียม จึงมีการ Normalized ด้วยจำนวนเอกสารผลลัพธ์สุดท้าย แสดงผลทั้งแบบ ก่อน และหลัง Normalized การทดลองในหัวข้อนี้จะใช้ข้อมูลจำนวนเอกสารที่ได้รวบรวมจากการคำนวณในโปรแกรมแมทแลป และจะนำข้อมูลนี้แทนจำนวน Posting ในเครือข่ายซึ่งมีขนาดอันละ 28 bytes ดังหัวข้อที่ 4.2.1

ตารางที่4.8 ผลการเปรียบเทียบปริมาณข้อมูลของดัชนีแบบเทอมเดียว

ปริมาณข้อมูล	แบบสุ่มลำดับ	แบบเรียงลำดับ
ผลรวม (Bytes)	32,211,788	14,237,888
ค่าเฉลี่ย/ คิวรี่ (Bytes)	236,851	104,690

ตารางที่4.9 ผลการเปรียบเทียบปริมาณข้อมูลของดัชนีแบบเทอมเซตที่เรียงและไม่เรียงลำดับคำค้นหาตามจำนวนเอกสาร

ปริมาณข้อมูล	แบบสุ่มลำดับ	แบบเรียงลำดับ
ผลรวม (Bytes)	743,596	401,632
ค่าเฉลี่ย/ คิวรี่ (Bytes)	5,467	2,953

ตารางที่ 4.10 ผลการเปรียบเทียบปริมาณข้อมูลของดัชนีแบบเดียวและแบบเทอมเซตหลัง Normalized

ปริมาณข้อมูล	แบบสุ่มลำดับ	แบบเรียงลำดับ	แบบเทอมเซต เรียงลำดับ	แบบเทอมเซต สุ่มลำดับ
ผลรวม (Bytes)	617.21	272.81	75.53	139.85

จากตารางที่ 4.15 ดัชนีแบบเทอมเซตช่วยให้ปริมาณข้อมูลลดลง 94.78% เปรียบเทียบกับปริมาณข้อมูลของดัชนีเทอมเดียวแบบที่เรียงลำดับคีย์เวิร์ด และลดลง 48.74% เมื่อทำการ Normalized ด้วยจำนวนผลลัพธ์แล้ว

4.2.3 โปรแกรมจำลองเครือข่าย peer to peer

โปรแกรมที่ใช้จำลองเครือข่าย peer to peer ในงานวิจัยนี้ได้แก่ PeerfactSim.KOM [11] พัฒนาโดย Technische Universität Darmstadt ประเทศเยอรมันในปี 2011 PeerfactSim.KOM จำลองการส่งข้อมูลระหว่าง peer โดย ใช้ฟังก์ชันทางคณิตศาสตร์ ตามข้อมูลงานวิจัยที่ทำการวัดข้อมูลอินเตอร์เน็ต pingER project และข้อมูล CAIDA Project [13] ในการจำลองตำแหน่งของ peer ตามสภาพภูมิศาสตร์ (GNP Grobal Network Positioning) สำหรับจำลอง latency, jitter และ packet lost

4.2.4 ค่าพารามิเตอร์ของโปรแกรมจำลองเครือข่าย

กำหนดค่าพารามิเตอร์ที่ต้องใช้ในแบบจำลองและระยะเวลาที่ใช้ในการจำลองเครือข่าย ในการจำลองนี้จะกำหนดให้เวลาเริ่มต้นค้นหาที่เวลา 20m ไปจนสิ้นสุด 120m การกระจายของโหนดจะใช้ข้อมูล GNP โดยกำหนดให้อยู่ภายในโซนเดียวกัน (Latin America) จำนวน 1000 โหนด

ตารางที่ 4.11 แสดงค่าพารามิเตอร์ที่ใช้ทดลองในแบบจำลอง

พารามิเตอร์	ค่าที่ใช้	คำอธิบาย
Overlay	Chord	ชนิด Overlay โพรโตคอลที่ใช้ในการจำลอง
Size	1000	จำนวนโหนด
Netlayer	GNP	จำลองการกระจายของโหนดตามตำแหน่งภูมิศาสตร์
StartTime	0m	เวลาเริ่มต้นการจำลอง (นาที)
FinishedTtime	120m	เวลาสิ้นสุดการจำลอง (นาที)

4.2.5 ปริมาณข้อมูล ของ Distibuted Hash Table (Chord)

แสดงปริมาณข้อมูลในเครือข่ายเพียร์-ทู-เพียร์แบบ Chord ทดลอง โดยกำหนดปริมาณข้อมูลของจำนวน posting เป็นศูนย์ เพื่อแสดงให้เห็นถึง overhead ของเครือข่ายแบบ Chord Overlay

ตารางที่ 4.12 แสดงปริมาณข้อมูลที่ใช้ในเครือข่าย ของโปรโตคอล Chord

Metrics	ปริมาณข้อมูลของ Chord (Bytes)
Average Bandwidth In (Bytes/sec)	276,183
Average Bandwidth Out (Bytes/sec)	276,186
Average Bytes Sent (Bytes)	2,071,395
Average Bytes Received (Byes)	2,071,378

4.2.6 ปริมาณข้อมูลของการค้นหาจากดัชนีเทอมเดียวแบบที่เรียงและไม่เรียงลำดับคำค้นหา เปรียบเทียบกับดัชนีแบบเทอมเซตโดยใช้โปรแกรมจำลอง

ทำการเปรียบเทียบการค้นหาจากดัชนีเทอมเดียวที่เรียงและไม่เรียงลำดับคีย์เวิร์ด เปรียบเทียบกับดัชนีแบบเทอมเซต โดยใช้โปรแกรมจำลองเครือข่ายเพียร์-ทู-เพียร์ ข้อมูลที่เป็นปริมาณ payload คือจำนวนเอกสาร x ขนาดโครงสร้างดัชนี 28 bytes โดยใช้ข้อมูลและคิวรีในชุดที่ 1 จากตารางที่ 4.18 และ 4.19 แสดงให้เห็นค่าที่ลดลงของการใช้ดัชนีแบบเทอมเซต และการจัดลำดับแบบเรียงคีย์เวิร์ด เพื่อแสดงให้เห็นถึงความสอดคล้องกับผลการทดลองในหัวข้อที่ 4.2.2

ตารางที่ 4.13 ผลการเปรียบเทียบปริมาณข้อมูลของดัชนีแบบเดียวและแบบเทอมเซตก่อน Normalized

	ปริมาณข้อมูลเฉลี่ยก่อน Normalized		
ตัววัด	**ค้นหาจากดัชนีเทอมเดียวแบบไม่เรียงลำดับคีย์เวิร์ด**	**ค้นหาจากดัชนีเทอมเดียวแบบเรียงลำดับคีย์เวิร์ด**	**ค้นหาจากดัชนีเทอมเซต**
Average Bandwidth (Bytes/sec)	4,587,303	4,545,202	289,940
Average Bytes Sent (Bytes)	34,404,848	34,089,058	2,174,554

ตารางที่ 4.14 ผลการเปรียบเทียบปริมาณข้อมูลของดัชนีแบบเดียวและแบบเทอมเซตหลัง Normalized

	ปริมาณข้อมูลเฉลี่ยหลัง Normalized		
ตัววัด	**ค้นหาจากดัชนีเทอมเดียวแบบไม่เรียงลำดับคีย์เวิร์ด**	**ค้นหาจากดัชนีเทอมเดียวแบบเรียงลำดับคีย์เวิร์ด**	**ค้นหาจากดัชนีเทอมเซต**
Average Bandwidth (Bytes/sec)	11,197	11,094	10,455

Average Bytes Sent (Bytes)	83,979	83,208	78,418

บทที่5

บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

1) จากการทดลองในบทที่4 หัวข้อ 4.1.5 การทดสอบคิวรีแบบสามคีย์เวิร์ดโดยใช้ค่า IDF นั้น ไม่การันตีว่าจะได้ผลดีที่สุด แต่ผลที่ได้จะดีกว่าค่าเฉลี่ย หรือวิธีแบบสุ่มจับคู่ในทุกขนาดดัชนี

2) จากการทดลองในบทที่4 หัวข้อ 4.1.4 จากสูตรการพรุนนิ่งดัชนีเทอมเซต การค้นหาแบบจับคู่ด้วยเทอมเดียวนั้นมีโอกาสน้อยมากที่จะเจอเอกสารเนื่องจากเทอมเดียวถูกตัดออกจากดัชนีเป็นจำนวนมาก สอดคล้องกับผลการทดลองในหัวข้อที่ 4.1.6

3) การทดลองในบทที่4 หัวข้อ 4.1.5 การเลือกคีย์เวิร์ดที่จับคู่ให้ผลดีเป็นสัดส่วนที่มากขึ้นเมื่อขนาดดัชนีมีการพรุนนิ่งหรือคัดออกจำนวนมาก

4) การทดลองที่ในบทที่ 4 หัวข้อที่ 4.1.6 การเลือกจับคู่ดัชนีในรูปแบบที่มีเทอมเดียวด้วยนั้นให้ผลลัพธ์ที่สูงขึ้นเพียง 1 % แต่การค้นแบบสามคีย์เวิร์ดโดยรวมรูปแบบวิธีนี้ด้วยนั้น ต้องเพิ่มโอเปอร์เรชันในการเปรียบเทียบเพิ่มขึ้นอีกเท่าตัว

5.2 ปัญหาที่พบและข้อเสนอแนะ

1) การทำดัชนีแบบสามเทอมโดยเอาทุกเทอมมาแข่งขันกันตามสูตรในบทที่3 สมการที่ 3.4 ประเด็นที่เป็นปัญหาคือกรณีของเทอมเดียวที่งานวิจัยก่อนหน้าไม่ได้ระบุไว้อย่างชัดเจน ดังนั้น กรณีการค้นหาแบบเทอมเดียวจากดัชนีเทอมเซตที่มีการพรุนนิ่งตามสูตรนั้นมีโอกาสที่จะไม่เจอเอกสาร เพราะเทอมเดียวถูกตัดออกดัชนีเป็นจำนวนมาก ซึ่งในทางปฏิบัติแล้วไม่ควรเป็นเช่นนั้น การแก้ปัญหาในการค้นหาแบบหลายคีย์เวิร์ดที่จำนวนคำค้นหามีมากกว่าขนาดเทอมเซต จึงควรแตกคีย์เวิร์ดให้อยู่ในรูปของเทอมเซตทั้งหมดแล้วจึงทำการค้นหาและรวบรวมผลลัพธ์

2) การค้นหาแบบหลายคีย์เวิร์ดจากดัชนีเทอมเซตไม่สามารถเพิ่มขนาดเทอมเซตให้มีขนาดมากพอที่จะรองรับจำนวนคีย์เวิร์ดได้ เพราะในทางปฏิบัติมีข้อจำกัด ด้านเวลาการประมวลผลและพื้นที่ในจัดเก็บข้อมูล การจัดทำและค้นหาดัชนีเทอมเซตต้องอาศัยทรัพยากรในการประมวลผลและจัดเก็บที่สูงมาก คอมพิวเตอร์ทีที่ใช้ประมวลผลต้องมีประสิทธิภาพสูง

เอกสารอ้างอิง

[1] D. Gnawali, "A Keyword-Set Search System for Peer-to-Peer Networks," Master's thesis, MIT, 2002.

[2] H. Chen, J. H. Jin, Y. Liu, L. M. Ni, TSS: Efficient Term Set Search in Large Peer-to-Peer Textual Collections, *IEEE, Transactions on Computers*, vol. 59, no. 7, pp. 969-980.

[3] P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In *Proc. ACM International Middleware Conference*, pages 21–40, Rio de Janeiro, Brazil, June 2003.

[4] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[5] G. Salton and C. Buckley, "Term Weighting Approaches in Automatic Text Retrieval," Information Processing and Management, vol. 24, pp. 513-523, 1988.

[6] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregate Information," Proc. IEEE Symp. Foundations of Computer Science (FOCS), 2003.

[7] I. Stoica, R. Morris *et al.*, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Trans. Net.*, vol. 11, no. 1, 2003, pp. 17–32.

[8] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", proceedings of ACM SIGCOMM 2001, San Diego, CA, August 2001.

[9] R. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[10] INEX 2009 Collection, [Online]. Available: http:// www.mpi-inf.mpg.de/departments/d5/software/inex/

[11] PeerfactSim.KOM - A Large Scale Simulation Framework for Peer-to-Peer System, [Online]. Available: http://peerfact.kom.e-technik.tu-darmstadt.de/

[12] The Anatomy of a Large-scale Hypertextual Web Search Engine, [Online]. Available: http://infolab.stanford.edu/~backrub/google.html

[13] The Cooperative Association for Internet Data Analysis (CAIDA), [Online]. Available: www.caida.org

[14] H. Chen, H. Jin et al., "Efficient Multi-keyword Search over P2P Web" Chinese Web Innovations, Beijing, China, April 2008.

ภาคผนวก

ผลงานวิจัยที่ได้รับการตีพิมพ์

T. Jitnupong and N. Chotikakamthorn "An efficient P2P searching and indexing strategy for multi-keyword query" 5th Int. Conf. on Knowledge and Smart technplogy (KST), Chonburi, Thailand, 2013, pp.113-116

2013 5th International Conference on Knowledge and Smart Technology (KST)

An efficient P2P searching and indexing strategy for multi-keyword query

Tanapon Jitnupong and Noppom Chotikakamthorn
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
s2660410@kmitl.ac.th, noppom@it.kmitl.ac.th

***Abstract*—Multi-keyword search in a P2P repository system based on DHT suffers from high traffic cost due to intersection operation of distributed inverted lists. Indexing by a set of keywords (term set) is one possible solution to reduce the cost. However, as the number of terms for each set increases, the number of possible term sets increases tremendously. Index pruning partly solves the problem, with the cost of reduced recall rate. In this paper, a multi-keyword search strategy for use with the term-set index pruning technique is proposed to increase recall rate, while retaining the same index size required for the case of a two-keyword search. Experimental result supports the predicted improvement.**

Keywords- p2p system; multi-keyword query; term-set index

I. INTRODUCTION

There are generally two searching approaches in the P2P (peer to peer) system; 1) federated search using unstructured P2P, and 2) distributed global inverted index on top of structured P2P. For the first approach, documents are indexed locally and independently on each server peer or node. Search in unstructured P2P needs flooding a query message to other peers. Query processing is computed locally on each peer. The results are sent back to a query-originating peer, where resulting aggregation takes place. For the second approach, a DHT [5, 6] is used for indexing and locating inverted-index. DHTs guarantee perfect recall and are able to locate matches within a small number of nodes. However DHTs may demand significant bandwidth for publishing the content and executing complicated searches, such as those with multiple keywords [1].

While a multi-keyword query is commonly supported by a centralized document/web search engine, supporting such feature in P2P systems remains a challenging issue. Existing P2P retrieval mechanism generally use DHT to independently map each keyword in a query to a set of document/node across the network that contain such keyword. Merging results for each keyword creates a lot of traffic. To reduce such cost, a term-set index can be pre-computed to map a set of terms to the list of global documents containing them [2].

A term-set index is effective to reduce the communication cost [4] by avoiding the distributed intersection operation across a network. However, such approach can increase the index size significantly when a set of three or more keyword terms needs to be indexed.

To reduce an index size, Chen [2] proposes an index pruning method. With this method, only relevant documents/ term sets are retained. The trade-off is that a recall rate is reduced. Also, in practice, only term-sets of the small number of keywords may get indexed to avoid an excessively large index. For example, for three or more keywords, the number of term sets grows significantly. When an index of two-keyword term-sets is used for a query of three or more keywords, query keywords must be split into multiple keyword pairs for independent searches. As will be shown later, splitting keywords inappropriately can further reducing a recall rate.

Due to this shortcoming, we present in this paper an efficient term-set based index searching strategy with a two-term index applied to queries containing three or more keywords. Experiment was performed using the CISI collection data set [11] to evaluate the performance of the proposed method.

II. RELATE WORK

There are several researches on a multiple keyword search in a structured P2P network. Reynold and Vahdat [6] propose a search technique using a Bloom filter [7], Instead of the list document-ID, bit array is used to reduce search cost. Chen [3] optimizes a bloom filter to reduce false positive rate by pre computing statistical information of keyword. Gnawali [4] partition index by a set of keyword to reduce the number of connection nodes. Similarly, Clements [10] use term-sets for indexing metadata. However, an enormous index size make a simple implementation for a content file impractical. To reduce a term-set index size, Chen [2] proposed the TSS (Term-Set Search System) system which applies an index pruning method. The method utilizes TFxIDF [9] weighting scheme to rank the relevance between each term-set for an indexing document. TSS also utilizes Gossip protocol [8] to propagate the statistical information to all nodes in the system before publishing relevant term sets onto a global index, which is layered on top of DHT.

978-1-4673-4853-9/13/$31.00 ©2013 IEEE

Generally, a term-set index cannot include term-sets generated from all combinations of terms. Doing so will increase an index size significantly. With the index being restricted to only, says, two-keyword term-sets, a query of three or more keywords needs to be split into queries with smaller number of keywords. For example, let A, B and C be the keywords in a query to be processed with a two-keyword term-set index. The three query keywords need to be split into two keyword pairs for two independent searches to be performed. The results are then aggregated. There is more than one possibility for such keyword-split and aggregation operation, e.g., AB∩AC, AB∩BC, and AC∩BC. Without index pruning, any choice of such operation yields the same result. However, with a pruned index, this is not the case. For example, if A is a bad keyword (in a sense that it is a generic term and thus gives low TFxIDF value), with the choice of AB∩AC operation, some relevant documents containing stronger keywords of B and C may not appear in an aggregated result. In this case, more relevant documents are expected from an aggregated result obtained by using either AB∩BC or AC∩BC as a keyword-split and aggregation operation. In this paper, a method for choosing an appropriate keyword-split and aggregation operation is described. With the proposed scheme, noticeable improvement in terms of recall rate is obtained as compared with a method based on random selection among possible choices of the operation.

III. Multi-Keyword Search using a Term-set Based Index

A. Term set generating

Term set is a combination of terms in a document. For example, let A, B and C be the words in a document d. The generated index entries will be <AB, d>, <AC, d>, and <BC, d>. In this paper, the number of keywords for each term set is limited to two, to avoid excessive increase of index size. To generate possible term sets, single keywords are first extracted from documents in a collection. All possible pairings of the extracted single keywords, whose number is denoted by n, are used as two-word term sets (of which number is $C(n, 2)$).

B. Term set pruning

After a term-set list is obtained, index pruning is performed using TF-IDF as follows:

1) Compute the Term-Frequency (TF): A weight of a keyword t for each document d is computed from a Term-Frequency (TF) by the following equation.

$$w_{d,t} = TF_t = 1 + \log(f_{d,t}) \quad (1)$$

where $f_{d,t}$ is the number of times that term t appears in document d.

2) Compute the Inverse Document Frequency (IDF): For each keyword t by using certain global document statistical data. Such information is propagated by the Gossip protocol. The data consists of the number of documents in the collection (N), and the number of documents containing the term t (f_t). Calculation is performed by using the following equation.

$$w_{s,t} = IDF = \log(1 + \frac{N}{f_t}) \quad (2)$$

3) Compute a similarity: (or correlation) between the document d and a term set s by (3).

$$sim(s,d) = \frac{\sum_{t \in s} w_{s,t} \times w_{d,t}}{\sqrt{|s| \times |d|}} \quad (3)$$

From Eq. (3), with a chosen pruning parameter λ, a term set s is discarded from the index if its similarity value is ranked below the top $\lambda n \log(n)$ term sets.

Algorithm: Index generating and pruning

input: document vector
output: pruned term-set vector
  for each document d in collection
  $n \leftarrow$ number of terms in d
  $s \leftarrow$ combination of terms in d $(C(n,2))$
    for each term-set s
      $w_{d,t} \leftarrow \{TF_{t1}, TF_{t2}\}$
      $w_{s,t} \leftarrow \{IDF_{t1}, IDF_{t2}\}$
      $sim_{s,d} \leftarrow sim(w_{d,t}, w_{s,t})$
    end for
    rank$\{sim_{s,d}\}$ in a descending
    truncate $\{sim_{s,d}\}$ at position that less than top $\lambda n \log(n)$
  end for

Figure 1. Index generating and pruning algorithm.

C. Keyword search strategy

As explained before, when a P2P node receives a query containing three or more keywords, a keyword-split and aggregation operation needs to be performed. The proposed multi-keyword search strategy is related to the keyword-split part of the operation. The method is explained below.

1) Choosing a representative keyword: Here, a representative keyword is a keyword that is less commonly found in other documents. By using the TFxIDF weighting scheme, it is a keyword that has high TFxIDF value. Therefore, any term-set which contains such keyword is likely to be preserved during pruning. Keyword splitting is performed as follows:

a) Getting the score: For each keyword in a query, look for (or calculate) its TF and |d| values from the local index. Then compute TF/|d| and rank the result in descending order.

b) Keyword selection: Select only the first 10% of the ordered result from the previous step. Compute the average

value from the selected TF/|d|. Choosing a keyword which has the highest averaged TF/|d| as a representative keyword.

c) Keyword splitting: A representative keyword must appear in any keyword pair as a result of keyword splitting. For example, if a query contains keywords A, B, C, with C being the representative keyword, the query is split into CA and CB respectively.

2) Searching and aggregating the result: When a keyword-split operation is completed, the search and aggregation is performed as follows:

a) Searching: For each query (keyword-split), the keyword are sorted and hashed to be a key, the lookup service (DHT) will use a key to fetch a document on a term-set index. After that, the document of each query will have been aggregated by the intersection operation.

b) Result ranking: For each document result, combining a TF-IDF value of each keyword and compute a similarity score to rank documents by their relevance.

IV. EXPERIMENTALS

In this section, results from the evaluation of the proposed multi-keyword search strategy are reported. The experiments were conducted using the CISI [11] collection of information science abstracts. The contents consist of 1460 documents and 5540 terms.

For the query work load, we cut stop-word and selected the relevant 3-5 keywords based on Term Frequency (TF) from 1460 documents. Examples of queries used in the experiments are: "distribution efficiency indexing", "retrieval analysis effectiveness". The metric used to evaluate search performance are recall and precision.

In the experiments, documents were randomly chosen and assigned to each node, from where a local index was generated. The search performance was compared with that of a simple keyword-splitting method, which arbitrarily or randomly performs a keyword-splitting operation. The averaged value of results from all possible random keyword-splitting patterns was used to represent the result from the simple keyword-splitting method (see Section II for an illustrative example). Recall and precision rates were computed using the following equations.

$$\text{recall} = \frac{|\{\text{relevant docs}\} \cap \{\text{retrived docs}\}|}{|\{\text{relevant docs}\}|} \quad (4)$$

$$\text{precision} = \frac{|\{\text{relevant docs}\} \cap \{\text{retrived docs}\}|}{|\{\text{retrived docs}\}|} \quad (5)$$

In the second experiment, 163 queries were randomly selected from 1460 queries, each of which yields no fewer than ten retrieved document items. The precision rate was computed from the top-ten ranked retrieval results. The performance of the proposed method was compared with that of the TSS method.

The TSS term-set size limit is 3. We used three terms in a query for comparing the TSS method with our method. This Experiment shows that our approach has higher Recall at all index size in Fig 2. Also, as shown in Fig.3, our approach has higher precision rate.

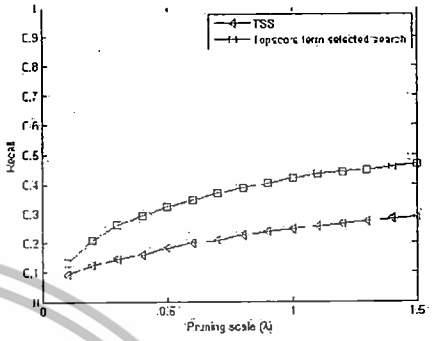


Figure 2. Recall rate comparison.

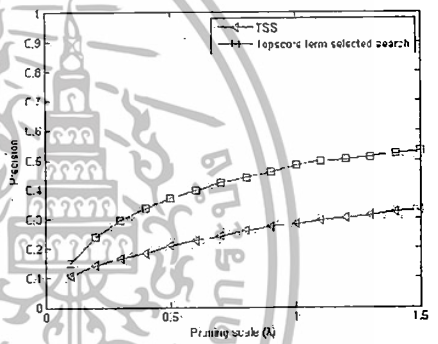


Figure 3. Precision comparison @top10 result.

V. CONCLUSION

In this paper, a search strategy for processing a query of three or more keywords, using a two-keyword term-set index has been described. The method utilizes TF-IDF weighting scheme. For the query of three or more keywords, a keyword-split and aggregation operation must be performed. A strategy for a keyword-split part of the operation that yields improved recall rate has been described. Experimental results confirm the improvement gained by the proposed method.

REFERENCES

[1] B. T. Loo *et al.*, "The Case for a Hybrid P2P Search Infrastructure," *Proc. 3rd Int'l. Wksp. Peer-to-Peer Systems (IPTPS)*, San Diego , California, USA, Feb. 26–27, 2004.

[2] H. Chen, J. H. Jin, Y. Liu, L. M. Ni, TSS: Efficient Term Set Search in Large Peer-to-Peer Textual Collections, *IEEE, Transactions on Computers*, vol. 59, no. 7, pp. 969-980.

[3] H. Chen, H. Jin, J. Wang, L. Chen, Y. Liu and L.M. Ni, "Efficient Multi-Keyword Search over P2P Web," Proc. Int'l World Wide Web Conf. (WWW), 2008.

[4] D. Gnawali, "A Keyword-Set Search System for Peer-to-Peer Networks," Master's thesis, MIT, 2002.

[5] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", proceedings of ACM SIGCOMM 2001, San Diego, CA, August 2001.

[6] P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In *Proc. ACM International Middleware Conference*, pages 21–40, Rio de Janeiro, Brazil, June 2003.

[7] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[8] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregate Information," Proc. IEEE Symp. Foundations of Computer Science (FOCS), 2003.

[9] G. Salton and C. Buckley, "Term Weighting Approaches in Automatic Text Retrieval," Information Processing and Management, vol. 24, pp. 513-523, 1988.

[10] A. T. Clements, D. R. K. Ports, and D. R. Karger. "Arpeggio : Metadata searching and content sharing with Chord." In Proc. IPTPS '05, Ithaca, NY, Feb. 2005.

[11] http://web.eecs.utk.edu/research/lsi/corpa.htm

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายธนพล จิตนุพงศ์
วันเดือนปีเกิด	10 กุมภาพันธ์ 2530
ที่อยู่	121/4 ม.4 ต.ท่าซัก อ.เมือง จ. นครศรีธรรมราช 80000
ประวัติการศึกษา	2551-อส.บ. เทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ