การทำนายการเรียกใช้ข้อมูลของ Web Server

Predicting Requests to Web Servers

โดย

นาย กิตติศักดิ์ ว่องไว

รหัส 42067175

อาจารย์ที่ปรึกษา

ดร. โชติพัชร์ ภรณวลัย

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 1 ปีการศึกษา 2544
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การทำนายการเรียกใช้ข้อมูลของ Web Server
นักศึกษา	นายกิตติศักดิ์ ว่องไว
อาจารย์ที่ปรึกษา	ดร.โชติพัชร์ ภรณวลัย
ระดับการศึกษา	วิทยาศาสตรบัณฑิต สาขาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2544

บทคัดย่อ

การเติบโตของ Internet ทำให้การใช้บริการบนเครือข่ายเพิ่มขึ้นอย่างมาก ดังนั้นการใช้บริการช่องข้อมูลบนเครือข่ายในการใช้ Web Pages ก็เพิ่มมากขึ้นเช่นกัน การที่ Server ให้ Hint แก่ฝั่ง Client ทำการ Caching และ Prefetching นั้น จะช่วยลดปัญหาดังกล่าวได้ โดยใช้วิธีการจัดกลุ่มทรัพยากรที่มีความน่าจะเป็นว่าใช้บริการพร้อมกันให้อยู่ใน Volume เดียวกัน ซึ่งจะช่วยในการปรับปรุงโปรแกรม Prefetching , Cache Replacement และ Cache Validation ในรายงานนี้จะกล่าวถึงหลักเกณฑ์การสร้างกลุ่มที่เหมาะสม และการปรับปรุงประสิทธิภาพ Heuristic การปรับค่าตัวแปรต่างๆใน Algorithm ที่ใช้ทำนายการใช้บริการนั้น เพื่อลดคำนายที่ผิด และจำกัดขนาดของ Hint โดยจะทำการวิเคราะห์จาก Server Logs ขนาดใหญ่ โดยจะหารูปแบบการใช้บริการเพื่อสร้างและประเมิน Volume เพื่อจะแสดงว่าสามารถให้ผลการทำนายที่ลงทุนต่ำแต่มีความแม่นยำของผลทำนาย

Title	Predicting Requests to Web Servers
Student	Mr. Kittisak Wongwai
Advisor	Dr. Chotipat Panavalai
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2001

ABSTRACT

The rapid growth of internet has led to dramatic increase in internet traffic, as well as user perceived latency in retrieving web pages has increased. Caching and prefetching at the client side, aided by hints from the server, are attempts at solving this problem. We suggest techniques to group resources that are likely to be accessed together into volumes , which are used to generate hints tailored to indivual applications , such as prefetching , cache replacement , and cache validation . We have studied theoretical aspects of optimal volume construction , develop efficient heuristics and tune parameters of algorithm to predict the possible access while reducing false predictions and limiting the size of hints . We analyze a collection of large server logs to find access patterns to construct and evaluate volume at low cost with a high degree of precision

กิตติกรรมประกาศ

โครงการพัฒนาระบบงานเรื่องการทำนายการเรียกใช้ข้อมูลของเว็ปเซิร์ฟเวอร์ในฉบับนี้ ผู้เขียนขอขอบพระคุณ ดร. โชติพัชร์ ภรณวลัย อาจารย์ที่ปรึกษาที่ได้กรุณาให้คำปรึกษาและแนะนำ และขอขอบคุณผู้ที่เกี่ยวข้องทุก ๆ ท่านที่กรุณาให้คำปรึกษาและแนะนำวิธีการแก้ปัญหาต่าง ๆ ให้สามารถแก้ปัญหาให้สำเร็จลุล่วงไปได้

กิตติศักดิ์ ว่องไว

กันยายน 2544

สารบัญ

หน้า

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมา

ความนิยมของ World Wide Web เพิ่มมากขึ้น ทำให้การใช้บริการก็มากขึ้นตาม โดยการใช้บริการส่วนใหญ่จะใช้บริการ Web Page ซึ่งอาจจะมาจากหลายที่ หรือมีการใช้ข้อมูลประเภทอักษร, รูปภาพ และ Applets ข้อมูลที่ได้จาก Web Server นั้นจะส่งผ่านเครือข่าย ยกเว้นกรณีที่มีข้อมูลอยู่ใน Browser Cache หรือที่ Proxy Cache ในการเรียกข้อมูลนั้น จะต้องสร้างการเชื่อมต่อ TCP และใช้ URL ที่เหมาะสมแล้ว Server จึงจะส่งข้อมูลได้ การเปิดบริการเชื่อมต่อของ TCP , การ Summit ของ Request และการส่งการตอบรับ จะทำให้เกิดการทำงานที่อยู่เบื้องหลังและใช้ทรัพยากรของเครือข่ายของเส้นทางจาก Server กับ Client ดังนั้นการเก็บข้อมูลที่ได้รับความนิยมไว้ใกล้ฝั่ง Client หรือ ใน Proxy Caches ทำให้ลดการใช้บริการของเครือข่าย

การใช้ Proxy เกี่ยวกับการจัดการทรัพยากรและ การเชื่อมต่อ TCP นั้นจะขึ้นกับสมมุติฐานเกี่ยวกับรูปแบบการใช้บริการของ Client เช่น การใส่ข้อมูลใน Cache นั้นจะขึ้นกับการชี้ว่าข้อมูลใดมีแนวโน้มที่จะนำมาใช้ในอนาคต เช่นเดียวกัน การ Prefetching ก็ขึ้นกับการทำนายการเรียกใช้บริการในอนาคต Server ซึ่งเป็นผู้ให้บริการแก่ผู้ใช้บริการและ Proxy นั้น จะทำการประเมินว่าข้อมูลใดได้รับความนิยมและข้อมูลใดที่น่าจะถูกเรียกใช้ถัดมาซึ่งจะทำให้ช่วยลดเวลาที่ใช้บริการให้น้อยลงได้

ในการศึกษานี้จะกล่าวถึงการออกแบบและประเมินผล Algorithm ของการจัดกลุ่มข้อมูล (Resources) ที่มีความน่าจะเป็นว่าการใช้บริการที่พร้อมกันจะให้อยู่ด้วยกัน โดย Algorithm นี้จะต้องประเมินได้ว่าข้อมูลใดจะถูกเรียกใช้ในช่วงเวลาในอนาคต และ การปรับค่าตัวแปรของ Algorithm เพื่อจัดการความสมดุลระหว่าง ผลทำนายที่เป็นไปได้ กับการควบคุมจำนวนความผิดพลาด พร้อมทั้งจำกัดจำนวน ของ Hint ที่จะส่งไปให้ Proxy ดังนั้นเราจะพิจารณาถึงวิธีการที่จะลดการคำนวณในการสร้างและประเมินของ Volume นั้น โดยในการศึกษานี้จะกล่าวถึงสิ่งต่อไปนี้

Processing on Web Server Log

เสนอวิธีการ Cleaning ของ Server Log และสร้างรูปแบบที่เหมาะสมสำหรับการทำงานถัดมา

Problem Formulation

เสนอถึงสูตรทางทฤษฎีที่แม่นยำของปัญหาในการสร้าง Volume ซึ่งอยู่บนช่วงเวลาหนึ่งสำหรับการเรียกใช้จากการทำนาย และ เกณฑ์ที่สำคัญ 3 อย่าง ซึ่งจะนำมาใช้ในการพิจารณาถึงหลายๆ ปัญหา

Greedy Heuristic

หลังจากพิสูจน์ว่า ปัญหานั้น NP-complete แล้ว จะใช้ Greedy Algorithm สร้าง Volume โดย Greedy Algorithm นี้สร้างจากแนวคิดเรื่อง Marginal Utility เพื่อจะใช้ในการเพิ่ม Resource ใน Volume จนกระทั่งถึงค่าที่กำหนดไว้ ซึ่ง Volume จะให้ถึงข้อมูลปัจจุบันใน Volume นั้น ถึงแม้ว่า Volume จาก Greedy นี้จะดีแต่จะใช้เวลาในการคำนวณดังนั้นเราจะพิจารณาอีกทางเลือกที่เร็วกว่า เช่น Pairwise ในการสร้าง Volume

Pairwise Volume

วิธีของ Heuristic ที่ขึ้นกับความเป็นไปได้ที่ว่า คู่ของข้อมูลนั้นจะถูกเรียกใช้ด้วยกันโดยอาศัย Probability Threshold เป็นตัวตัดสินว่าข้อมูลใดควรจะอยู่ใน Volume พร้อมทั้งกล่าวถึงประสิทธิภาพของ Algorithm ในการหาความเป็นไปได้ในการเรียกใช้ และ เทคนิคอื่นๆ ในการแสดงถึง Hint ที่ไม่มีประสิทธิภาพ ซึ่งไม่ควรสร้างการทำนายออกมา

Reduction of processing Requirements

การทำงานกับ Server Logs ขนาดใหญ่นั้นต้องการทรัพยากรในการคำนวณอย่างสำคัญ ดังนั้นจึงเป็นไปได้ที่จะลดข้อมูลในการคำนวณลงในการสร้าง Volume โดย Overhead ในการสร้างและประเมินค่า Volume นั้นสามารถลดลงด้วย การใช้ตัวอย่างของ Server Log การสร้าง Volume จากช่วงเวลาที่น้อยกว่า หรือจากบางส่วนจาก Client กับ Proxy นั้น จะเปรียบได้กับจากข้อมูลทั้งหมดของการเรียกใช้

1.2 วัตถุประสงค์ของโครงงานการพัฒนาระบบงาน

1. เพื่อศึกษาหาแนวทางที่เหมาะสมที่ใช้ในการทำนายข้อมูลที่คาดว่าจะถูกเรียกใช้ในอนาคต
2. ทำการทดลองเปรียบเทียบวัดผลเพื่อปรับปรุงประสิทธิภาพ และหาวิธีการที่เหมาะสม
3. เพื่อเพิ่มความเข้าใจในการออกแบบและประยุกต์ใช้อัลกอริทึมในการทำนาย
4. เพื่อเป็นแนวทางในการออกแบบและพัฒนาเครื่องมือสำหรับการทำคาต้าไมนิ่งจาก Web Server Log ในแบบอื่นๆ ต่อไป

1.3 แนวทางการศึกษา

1. ศึกษาแนวทางการนำดาต้าไมนิ่งมาประยุกต์ใช้กับการวิเคราะห์ข้อมูลที่ได้จากการเก็บประวัติการเข้าถึงเว็บ
2. กำหนดขอบเขตของการทำงาน
3. ศึกษาอัลกอริทึมในการจัดกลุ่มข้อมูล วิเคราะห์ความเป็นไปได้และพิจารณาตัดสินใจเลือกอัลกอริทึมที่เหมาะสมในการทำงาน
4. พัฒนาโปรแกรมโดยใช้ Borland C , GNU C Complier , Oracle8i และ PL/SQL และทดสอบ

1.4 ขอบเขตโครงการ

พัฒนาเครื่องมือเพื่อใช้ในการศึกษาการจัดกลุ่มข้อมูลตามคำทำนายการเรียกใช้เว็ปเพจ โดยตัวโปรแกรมมีลักษณะดังนี้

1. จัดการทำความสะอาดข้อมูลที่ได้จากการเก็บประวัติการใช้เว็ปเพจ
2. โปรแกรมจัดการกับข้อมูลล็อกไฟล์ตามรูปแบบที่กำหนดไว้เท่านั้น
3. ผู้ใช้งานสามารถนำข้อมูลผลลัพธ์ที่ได้ไปวิเคราะห์เพื่อนำไปใช้ประโยชน์ต่อไปด้วยตนเอง

1.5 ประโยชน์ที่คาดว่าได้รับ

1. ได้เรียนรู้และเข้าใจแนวคิดวิธีการที่นำมาใช้ในการจัดกลุ่มข้อมูลที่มีความน่าจะเป็นว่าจะถูกเรียกใช้บริการด้วยกัน
2. สามารถพัฒนาโปรแกรมที่ใช้ในการทำนายการเรียกใช้ข้อมูลให้กับ Web Server ได้
3. สามารถนำเอาข้อมูลสารสนเทศที่มีอยู่แล้วในระบบมาใช้ประโยชน์ได้มากขึ้น

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีเกี่ยวข้องในการศึกษานี้จะกล่าวถึง การจัดเตรียมข้อมูล การออกแบบและอัลกอริทึมของการจัดกลุ่มข้อมูล โดยสามารถประเมินผลได้ว่าอัลกอริทึมที่นำมาใช้นั้นได้

2.1 ปัญหาในการจัดกลุ่มข้อมูล (Volume Construction Problem)

ในส่วนนี้จะกล่าวถึง แบบจำลองในการทำนาย การเรียกข้อมูลในอนาคต และ พิจารณาถึงเกณฑ์เหมาะสมที่ใช้พิจารณา

2.1.1 แบบจำลองการทำนาย (Prediction Model)

การศึกษาถึงรูปแบบในการเข้าถึงข้อมูลที่เรียกใช้ในช่วงเร็วๆนี้ ซึ่งจะนำไปใช้ในการทำนายว่า ข้อมูลใดจะถูกเข้าถึงในอนาคตใกล้นี้ การโต้ตอบระหว่าง การ Request และ Response นั้นสามารถที่ถูกทำนายได้จาก Server โดยทำนายว่าข้อมูลใดที่ Client จะเข้าถึงในอนาคตนี้ รูปแบบของการเข้าถึงนี้ได้จาก Hypertext link ที่ถูกใช้ในช่วงที่ผ่านมาของ Web Pages เราจะพิจารณาถึงวิธีการที่สร้าง Hint โดยอาศัยการเรียกใช้ของปัจจุบันมากกว่าการเก็บข้อมูลการเข้าถึงของเก่าที่ผ่านมา เพราะการที่ Proxy จำนวนมากเรียกข้อมูล และ ติดต่อ Server นั้นจะใช้ Overhead ที่มาก

Server มีข้อมูลของการเข้าถึงที่ผ่านมาซึ่งคือ (s,a,t) หมายถึง Client s ร้องขอข้อมูล (Resource) a ที่เวลา tโดยข้อมูลเหล่านี้ได้จาก Server Log และ คำนวณก่อนเพื่อหาความสัมพันธ์ของ Resource ที่เกี่ยวข้องกัน เพื่อลด Overhead นี้ การคำนวณนี้สามารถทำที่ฝั่ง Client หรือทำในช่วงเวลาที่น้อยกว่า Volume สามารถถูกคำนวณเป็นประจำเพื่อแน่ใจว่า Server สร้าง Hint ที่ขึ้นกับข้อมูลเร็วๆ นี้จาก Logs แต่ละ Resource a มี Volume V_a ซึ่งจะมีรายการของ Resource ที่เป็นไปได้ที่จะถูกเรียกใช้ในอนาคตถัดมาหลังจากการเข้าถึง a ซึ่ง (s,a,t) นำมาใช้เพื่อให้เกิดผลลัพธ์และมีประสิทธิภาพของ Algorithm ของการคำนวณ $\{v_a\}$

คำทำนายของการเข้าถึง Resource b จะมีประโยชน์ถ้ามันเกิดในช่วงเวลาที่น้อยที่สุด τ และที่เวลามากที่สุด T ก่อนการเรียกใช้ ถ้าหากว่าคำทำนายสร้างมาในช่วงเวลาที่ใกล้การเข้าถึงของผลการทำนายนั้น ทำให้ Proxy ไม่มีเวลาเพียงพอในการดำเนินงานตาม Hint จาก Server เช่น Proxy อาจจะต้องดึงรูปภาพที่มากับ Page นี้ ในทางตรงข้าม คำทำนายที่ทำนายออกมาแล้วจะมีการเรียกใช้อีกในช่วงเวลาถัดไปนานมากนั้นควรจะถูกจำกัด โดยในทางปฏิบัติก็เป็นการยากอยู่แล้ว ที่จะทำนายถึงการเรียกใช้ข้อมูลในอนาคตที่ห่างไกล เพราะว่า Client มักจะเรียกใช้ข้อมูลที่เกิดขึ้นในช่วงเวลาที่

ผ่านมา ดังนั้น จะได้ว่า คำทำนายการเรียกใช้ Resource b ที่เวลา t นั้นจะเกิดเมื่อ Proxy ได้รับได้รับอย่างน้อย 1 Hint ของ b ในช่วงเวลา [t-T,t-τ)ทำนองเดียวกัน Hint ที่สร้างที่เวลา t ของ Resource b จะถูกต้องก็ต่อเมื่อ Proxy ได้รับการเรียกใช้งาน Resource b อย่างน้อย 1 ครั้ง ที่เวลา [t+τ ,t+T)

2.1.2 เกณฑ์ที่เหมาะสม (Optimization Criteria)

Hints ที่ส่งไปยัง Proxy นั้น ควรมีสัดส่วนของคำทำนายการร้องขอให้มากที่สุด ในขณะที่ Hint ที่ไม่ถูกต้อง ก็ควรมีสัดส่วน ที่น้อยที่สุด และ Resource ที่ให้หลาย Hint ในเวลาเดียวกันต้องทำให้เหลือเพียงคำทำนายเดียว Server ควรทำให้จำนวนของHint ให้น้อยที่สุดเพื่อลด Overhead ของการส่ง Hint และรับของ Proxy โดยเกณฑ์ที่เกี่ยวข้องเพื่อใช้พิจารณามี 3 ค่า ดังนี้

- Recall คือ ความเป็นไปได้ที่ Resource ที่จะถูกเรียกใช้นั้นควรสร้างให้ได้ อย่างน้อย 1 Hint ในช่วงเวลา [τ, T) วินาที ที่ผ่านมา
- Precision คือความเป็นได้ที่ Resource จากการทำนายนั้นควรถูกเข้าถึงจาก Client ภายในเวลา [τ, T) วินาที ถัดไป
-Average Hintsize คือ จำนวนเฉลี่ยของ Hints ในข้อมูลจาก Server ไป Proxy

เกณฑ์นี้บอกถึงปัญหาต่างๆ เช่น การทำให้ Recall มากๆ นั้นทำให้ถึงขอบเขตบนของ Hintsize , การทำให้ Hintsize น้อยๆนั้น มีผลทำให้ถึงขอบเขตล่างของ Recall , การทำให้ Recall มากๆนั้น มีผลทำให้ถึงขอบเขตล่างของ Precision และ การทำให้ Precision มากๆนั้นมีผลทำให้ถึงขอบเขตล่างของ Recall

การเลือกเกณฑ์ที่เหมาะสมนั้นขึ้นอยู่กับว่า Proxy ใช้ Server Hint อย่างไร เช่น งาน Prefecting นั้นต้องการ Precision มาก เพื่อหลีกเลี่ยงความเสียหายจากคำทำนายที่ผิด ในทางตรงข้ามงานของ Cache Coherency Policy นั้นต้องการ Recall มากเพื่อรับข้อมูลการเปลี่ยนแปลงส่วนใหญ่ของข้อมูลของ Cache การทำ Volume ให้เหมาะตามเกณฑ์นั้นจะเป็นลักษณะการได้อย่างเสียอย่าง คือ เมื่อพิจารณาที่ 2 เกณฑ์แรกนั้น จะเห็นว่า Recall เพิ่ม Hintsize ก็เพิ่ม หรือเมื่อพิจารณา 2 เกณฑ์หลังจะได้ว่า Precision เพิ่มในขณะที่ Recall ลดลง ซึ่งลักษณะคุณสมบัติแบบนี้จะเห็นได้จากการใช้ Greedy Volume

ปัญหาของความเหมาะสมนี้เป็นไปตาม NP-Complete ซึ่งประสิทธิภาพของการสร้าง Volume ที่ Server นั้นเสนอใช้ Heuristic ที่มีการคำนวณความน่าจะเป็นของ Pairwise ซึ่งคือการเข้าถึง Resource a นั้นจะต้องตามด้วย Resource b โดย Pairwise Heuristic จะทำการรวม b ไว้ใน V_a เมื่อค่าความน่าจะเป็นเกินค่าที่กำหนด ในขณะที่ Greedy Algorithm จะคำนวณซ้ำความน่าจะเป็นนี้อีกโดยขึ้นกับการตัดสินใจที่ผ่านมา และ Greedy Algorithm นี้ทำงานที่ ค่า O(log n) เมื่อ n เป็น

จำนวนของการร้องขอ ในขณะที่ Pairwise ทำงานที่อัตราส่วน $\theta(n)$ ในกรณีที่แย่สุด ในทางปฏิบัตินั้น Greedy Algorithm ไม่เกี่ยวกับ Pairwise Heuristic แต่ทั้งสองวิธีก็ให้ Precision และ Recall ที่สูง โดยเหมาะสมกับขนาดของ Hintsize แต่อย่างไรก็ตาม Greedy ก่อให้เกิด Overhead ในการคำนวณ ดังนั้น จึงต้องการใช้ Pairwise Heuristic

2.2 Greedy Algorithm

Greedy Heuristic อาศัยข้อมูลที่มีอยู่สร้าง Volume โดยทำ iterate เพื่อขยายเซ็ต $\{ V_a \}$ ซึ่งการทำ iterate แต่ล่ะครั้งดำเนินการกับ Server Log เพียงครั้งเดียวเพื่อประเมินหา Marginal Utility u(b,a) ของการเพิ่ม Resource b ใน V_a ซึ่งจะได้เซ็ตของ Volume ณ เวลานั้น

การกำหนด u(b,a) ขึ้นกับเกณฑ์ที่เหมาะสม เช่น เมื่อพิจารณาถึง Recall กับ Hintsize :

$$u(b, a) = \frac{\text{จำนวนของการเรียกใช้ b ซึ่งเป็นตามคำทำนายที่เพิ่มมา}}{\text{จำนวนของการเรียกใช้ a}}$$

Input :

A thredhold U

For each resource b , a set candidates(b)

Iterate :

Compute $u(b,a)$ for all b and a $\in$ candidates(b)

For every resource b :

Let a $\in$ candidates(b) be the resource that maximize $u(b,a)$

If $u(b,a) \geq U$, $v_a \leftarrow v_a \cup \{b\}$

Remove from candidates(b) all a with $u(b,a) < U$.

Until : $\forall b$, candidates(b) = $\varnothing$

รูปที่ 2.1 Greedy heuristic

จากรูปที่ 2.1 นั้น Volume สร้างจากค่า Threshold U ของ Marginal Utility โดยแต่ละ Resource b จะได้ค่าที่มากที่สุดของ $u(b,a)$ ถ้า $u(b,a) \geq U$ แล้วจะนำ b ใส่ใน v_a และ Algorithmจะสิ้นสุดเมื่อทุก $u(b,a) < U$ สำหรับทุก (b,a)

Algorithmจะเริ่มด้วย Volumeที่ว่าง โดยแต่ละรอบของ iterate แต่ละ Resource b จะถูกใส่เข้าไปอย่างมากที่สุด หนึ่ง v_a โดยแต่ละรายการใน v_a จะแบ่งตามค่า Marginal Utility ที่เกี่ยวข้อง

โดยแต่ละ Resource จะมีชุดของ Candidates(b) ซึ่งจะเป็นตัวให้คำทำนายนั้นจะประกอบด้วย Resource a ทั้งหมด ดังนั้นเวลาในการคำนวณของ Greedy Heuristic จะขึ้นกับ จำนวนครั้งของการ Iteration ซึ่งคือจำนวนของ Candidate Set (อย่างมากคือจำนวนของ Resource ทั้งหมด)

เนื่องจากว่าประสิทธิภาพของAlgorithm นี้สามารถปรับปรุงได้โดยจำกัดจำนวนเริ่มต้นของ Candidate Set ซึ่งการเริ่มต้นด้วยการให้ ทุก Resource เป็นเซ็ตของ Candidate(b) นั้น พบว่า การคำนวณจะเป็นกำลังสองตัว Candidates ดังนั้นเราเริ่มต้นด้วยการใช้เฉพาะคู่ที่มีค่าเริ่มต้น $u(b,a) \geq U$ ซึ่งพบว่าประสิทธิภาพจะดีขึ้นสำหรับการสร้างแบบ Pairwise Volume และเมื่อค่า Marginal Utility u(b,a) ไม่ได้เพิ่มค่าดังนั้น เราจึงนำบาง Resource ออกจากการเป็น Candidate (b) ได้เมื่อ $u(b,a) < U$

2.3 การสร้างกลุ่มข้อมูลด้วยวิธี Pairwise (Constructing Pairwise Probabilities)

จาก Greedy Heuristic สร้าง Volume ที่ดีแต่ก็ใช้เวลาในการคำนวณที่สูง ในตอนนี้เราจะมาพิจารณาวิธี Pairwise พร้อมทั้งวิธีการลดขนาด Volume โดยยังให้คำทำนายที่มีประสิทธิภาพ

2.3.1 Pairwise Counters

เมื่อลดความซับซ้อนของ Greedy Algorithm เราสร้าง Volume ที่ขึ้นกับความน่าจะเป็นที่ Resource a และ b จะถูก Access ด้วยกันซึ่งไม่ขึ้นกับ Resource อื่น โดยมีค่าที่พิจารณาดังนี้

- $P_{b|a}$ คือ ความน่าจะเป็นที่การ Request a จะตามด้วย การ Request b โดย Client เดิมในช่วงเวลา [τ, T) วินาที
- P_t คือ Probability Threshold โดยที่ Resource b จะใส่ใน v_a ถ้าค่า $P_{b|a}$ มากพอหรือเท่ากับค่า P_t ซึ่งค่านี้ขึ้นกับการพิจารณาถึงเกณฑ์ Precision , Recall และ Hintsize
- $C_{b|a}$ คือ Pairwise Counters หมายถึงจำนวนครั้ง access a ทีตามด้วย b ในช่วงเวลา [τ, T) วินาที
- C_a คือ Individual Counters หมายถึง จำนวนครั้ง access ของ resource a

โดยที่ $P_{b|a} = C_{b|a} / C_a$

รูปแบบต่างๆ ที่ต้องพิจารณา เช่น

- "abbb" เมื่อ b ถูก Request หลายครั้งหลังจากการ access a เราจะคิดว่า $C_{b|a}=1$ ถึงแม้ว่า C_b จะเพิ่มขึ้นในแต่ละครั้งการ access ของ b เพื่อไม่ให้ $C_{b|a}/C_a$ มากกว่าหนึ่ง หากสร้างการทำนายที่ผิด ก็เหมือนการขยายค่า ความน่าจะเป็นของ $P_{b|a}$ เช่น ลักษณะการ Request เป็น "abbbaa" ควรได้ว่า $P_{b|a}=1/3$ ไม่ใช่ 3/3
- "aaab" การซ้ำของ Request a นั้นต่างกับกรณีข้างบนโดยเราจะให้ Credit ว่า Resource a ให้คำทำนาย b ดังนั้นจะได้ว่า $P_{b|a}=3/3$ ไม่ใช่ 1/3

Counter $C_{b|a}$ และ C_a นั้นสามารถนับได้จาก การ Request ใน Server Log โดยความน่าจะเป็นของการคำนวณนั้นขึ้นกับค่า (s,a,t) หมายถึง Client s ทำการ Request a ที่เวลา t ซึ่ง Algorithm นับ Request ของ Client นั้นในเวลานั้น สำหรับแต่ละ Client ใช้ FIFO เก็บการ Request จากเวลา T วินาทีก่อนหน้านั้น ในการพิจารณา request b ลำดับแรก แล้วนำเอา request ที่ปรากฏมากกว่า T วินาทีออกไปจากหัวของ FIFO แล้วนับจากท้ายของรายการเพื่อหา $C_{b|a}$ สำหรับแต่ละ Resource a โดยข้าม Request ที่ปรากฏในช่วงเวลาน้อยกว่า τ วินาทีที่ผ่านมาซึ่งเรานับไปเรื่อยๆจนกว่าจะเจอ b ตัวใหม่ มิฉะนั้น Algorithm นี้จะให้ Credit แก่ Resource a ในการทำนาย resource b หลายครั้ง

เช่น $ax{b_1}y{b_2}$ ได้ว่า b_1 : เพิ่ม $C_{b|x}$, $C_{b|a}$

b_2 : เพิ่ม $C_{b|y}$ แต่ไม่เพิ่ม $C_{b|x}$, $C_{b|a}$

การใช้วิธี Pairwise ($P_{b|a}$) พบว่าจำนวน Counter เป็นกำลังสองของ $C_{b|a}$ ถึงแม้ว่ามีหลายคู่ของ Resource ที่แทบจะไม่เกี่ยวข้องกัน ความซับซ้อนของการนับนี้สามารถลดลง โดยการให้ Heuristic นี้สร้าง Counter เฉพาะคู่ที่น่าจะถูก access ด้วยกันเช่น

- Counter $C_{b|a}$ จะสร้างได้ ถ้า b เข้าถึงได้จาก a ที่ $h \geq 1$ ของ Hypertext Link
- Counter $C_{b|a}$ จะสร้างได้ เมื่อ URL ของ a และ b เหมือนกันในลำดับแรกๆ ที่ $k \geq 0$ ของ Directory
- การนับค่าจาก Log ในการสร้าง Counter $C_{b|a}$ ให้สร้างด้วยความน่าจะเป็นของ $C/(P_iC_a)$ โดย C คือค่าคงที่ขึ้นกับค่าระหว่างจำนวนทั้งหมดของ Counter ที่จะต้องสร้างและความเป็นไปได้ที่พลาดคู่ที่สำคัญ

2.3.2 Volume Thinning

การใช้ $P_{b|a}$ คำนวณเพื่อใส่ b ใน v_a นั้นจะไม่เกี่ยวข้องกับ Resource อื่นๆ เลยซึ่ง Resource นั้นอาจจะเป็นตัวทำนาย b ก็ได้ ดังนั้น การคำนวณในลักษณะนี้ทำให้ Precision ลดลงเช่น พบว่า 9

ใน 10 ของ Resource a ตามด้วย b คือ $P_{b|a}$=9/10 แต่พบว่าใน 9 ครั้งของ a ที่ตามด้วย b นั้นจะปรากฏพร้อมกับ Resource อื่น เช่น "xyab" ซึ่ง $P_{b|x}$, $P_{b|y} \geq P_t$ ถ้า v_x , v_y และ v_a ต่างก็มี Resource b จะได้ว่า resource a จะไม่มีการทำนาย b ออกมา เพราะ ได้ถูกทำนายโดย Resource อื่นก่อนหน้านี้แล้ว และยังพบว่าการใส่ b ใน v_a ยังเป็นการลด Precision ลง เพราะว่าจะมีค่าทำนายถึง b ผิด 1 ใน 10 ซึ่งลักษณะความผิดพลาดแบบนี้จะไม่ปรากฏใน Greedy Algorithm ซึ่งจะประเมิน b ว่าใส่ใน v_x หรือ v_y แล้วยังมีการประเมินกับ Resource อื่นอีก

เมื่อต้องการเพิ่ม Precision และลด Hintsize เราจะพิจารณาถึง Heuristic เพื่อคัดคำทำนายที่ไม่มีประสิทธิภาพออกไป โดยไม่ต้องคำนวณที่ซับซ้อนเหมือนกับ Greedy Algorithm โดยหลังจากหาค่า $P_{b|a}$ และ ใช้ P_t ในการสร้างชุดเริ่มต้น ก็จะทำการคำนวณครั้งที่สอง ผ่าน Log เพื่อหาประสิทธิภาพของแต่ละตัวทำนาย โดย Heuristic ต่างๆที่ใช้จะต่างกันที่การให้ Credit แต่ละตัวของตัวทำนายซึ่งมีหลายตัวที่นำหน้า b เช่น Heuristic ที่ให้ความสำคัญกับ Resource ตัวแรก ว่าเป็นผู้ทำนาย คือ xyab จะให้ Credit แก่ x เป็นผู้ทำนาย b แต่อีกวิธีตรงข้ามคือให้ความสำคัญกับ Resource ตัวท้ายที่ยังอยู่ในช่วง [τ, T) โดยอาศัยสมมติฐานที่ว่า การ access ณ เวลานั้น ต้องสัมพันธ์กับการ Request ที่ล่าสุดของ Client เดิม มากกว่า การ access ที่เกิดก่อน

Heuristic แบบนี้จะนับจำนวนของ $e_{b|y}$ (ที่ $P_{b|y} \geq p_t$) เพื่อใช้แสดงว่า Resource a จะได้รับ Credit ในการสร้างคำทำนาย b นั่นคือ Resource b จะใส่ใน v_a ถ้าอัตราส่วนของคำทำนายเกินค่า Effective Threshold e_t ($e_{b|y}$ /$C_{b|y}$ $\geq e_t$) ซึ่งหมายความว่า Resource b จะใส่ใน v_a ถ้า $P_{b|y}$ และ $e_{b|y}$ มีค่าสูงพอตามลำดับ ถึงแม้ว่า Heuristic นี้จะทำให้ได้ตัวทำนายที่มีประสิทธิภาพมันก็ยังลดขนาดของ Volume โดยไม่ต้องคำนวณความสัมพันธ์ระหว่าง resource ต่างๆ

ตัวอย่างเช่น Resource x,y และ z ปรากฏในรูปแบบ xyz และ yxz ซึ่ง Heuristic ทั้ง 2 แบบให้ credit ทั้ง x และ y อย่างละครึ่งในการทำนาย z ถ้ากำหนดค่า e_t ให้มากกว่า 0.5 ก็ทำให้ทั้ง v_x และ v_y ไม่สามารถใส่ z ใน Volume ได้ ถึงแม้ว่าการใส่ z ใน Volume ใด Volume หนึ่งนั้นจะให้คำทำนายที่ถูกต้องของการ access z หรือถ้า e_t น้อยกว่า 0.5 แล้วก็จะได้ว่าต้องใส่ z ในทั้งสอง volume ซึ่งจะทำให้ 1 ใน 2 คำทำนายนั้นฟุ่มเฟือย ในทางตรงข้าม Greedy Algorithm จะเลือก 1 ใน 2 resource (x หรือ y) เพื่อใส่ z แล้วยังประเมินกับ Resource อื่นอีกด้วย

2.4 การดำเนินการกับ Web Server Log (Processing on Server Web Server Log)

ขบวนการในการจัดการ Server Log ขนาดใหญ่ที่หลากหลายนั้น มีดังนี้

- Processing Overheads

โดยทั่วไป Server Log มีข้อมูลจำนวนมากที่เกี่ยวกับ Client, Request และ Resource ซึ่งทำให้ใช้การคำนวณและหน่วยความจำที่มากในการจัดการข้อมูล

- Data integrity

บางครั้งรายการข้อมูลใน Server Log อาจจะมีข้อผิดพลาด หรือ ความไม่สอดคล้องกันของข้อมูลซึ่งต้องมีการละเว้น หรือจัดการ นอกจากนี้ การ Request ไปยัง Resource หนึ่งๆ อาจจะมี URL หลายอัน ซึ่งจะต้องจัดแสดงให้รวมกัน

- Privacy and Security

โดยทั่วไป Server Log เปิดเผยถึงข้อมูลละเอียดเกี่ยวกับ Client ที่ทำการ Request และ Resource ที่ถูก Request เช่น เวลาในการแจกจ่าย และ ความถี่ของการ Request จาก Client , การใช้เวลาใน site นั้น และลักษณะ ของ Resource ที่จะถูก Request

จากประเด็นดังกล่าว จะเสนอกระบวนการ Cleaning ของ Server Log และ สร้างรูปแบบที่เหมาะสมสำหรับเพื่อการทำงานถัดมา ถึงแม้ว่าเราจะจำกัดการพิจารณาแต่เพียง Server Log แต่ ก็สามารถแก้ไขให้เหมาะสมแล้วนำไปใช้กับ Proxy หรือ Client Log ได้เช่นกัน

2.4.1 การทำความสะอาด Servers Logs

การทำงานของการ request ของ HTTP นั้น Web Server สร้างรายการ Log ที่ประกอบด้วย field ต่างๆ ซึ่งจำนวนของ field จะอยู่ในช่วงต่างๆ จนถึง 20 กว่าค่า ตามแล้วแต่ละ Server ซึ่งมีรูปแบบต่างๆ ที่แตกต่างกันของ โดยมีมากกว่าสิบแบบที่แตกต่างไปตามรูปแบบของ Log นั้นๆ แต่ในการทำการศึกษาในครั้งนี้อาศัยข้อมูล Web Server Log จาก www.kmitl.ac.th ซึ่งมีรูปแบบคือ Apache's ECLF (Extended Common Log Format) โดยมีลักษณะของ field ใน Log Apache HTTP Server Version 1.3 ดังนี้

" host ident authuser date request status bytes "

โดยที่ Log file จะแยกการ request ของ client ไว้ที่แต่ละบรรทัด ถ้าสัญลักษณ์ไม่ได้แสดงค่าใด ก็จะแทนด้วยเครื่องหมาย hyphen (-) ซึ่งความหมายของแต่ละค่ามีดังนี้

host แสดงถึง domain name ของ client หรือ IP ของมัน ถ้าไม่สามารถแสดงชื่อได้

ident ถ้า identity Check ได้ใช้งาน และ identd ทำงานที่เครื่องของ Client แล้วจะรายงานถึง ข้อมูล identity

authuser ถ้า request เป็น password เพื่อป้องกัน document แล้วจะเป็น userid ซึ่งใช้กับ request

date แสดง date และ เวลาในการ request ซึ่งมี format ดังนี้

date =[day/month/year:hour:minute:second zone]

day =2*digit

month =3*letter

year =4*digit

hour =2*digit

minute =2*digit

second =2*digit

zone =('+'| '-') 4*digit

request การ request จาก client ซึ่งอยู่ใน double quotes ("")

status 3 digit แสดงสถานะที่ส่งกลับไปยัง client

bytes จำนวนของ bytes ในวัตถุที่ส่งกลับไปยัง client โดยไม่รวม headers.

ถึงแม้ว่าโดยทั่วไป field เหล่านี้จะถูกกำหนดและพิมอย่างถูกต้อง แต่ละรายการก็อาจจะผิดพลาดได้ ซึ่งก็จะขึ้นกับความแข็งแรงของกลไกการสร้าง Log หรือ I/O ที่เกี่ยวข้อง ซึ่ง Log อาจจะมีความผิดพลาดใน field ที่ไม่สร้างโดย Server หรือรายการที่ผิดเนื่องจากการทำรายการของ Log ไม่มีการจัดการป้องกันที่ดีพอ

ข้อผิดพลาดที่มากมายนี้สามารถตรวจพบด้วยวิธีการ Programming ที่ป้องกันไว้ เช่น ในขั้นตอนการอ่าน Server Log จะทำการตรวจสอบว่า แต่ละรายการมีจำนวนของ field ตามที่ต้องการหรือไม่ เช่น ตรวจโดยการส่งค่ากลับมา ด้วย คำสั่ง scanf ซึ่งการตรวจรายการที่ผิดพลาดทำด้วย manual ถึงแม้ว่าส่วนใหญ่ของรายการที่ไม่ถูกต้องนั้นจะถูกลบออกไป แต่บางกรณี URL String มีอักขระขึ้นบรรทัดใหม่ซึ่งทำให้ scanf ตรวจจุดสิ้นสุดของการขึ้นบรรทัดใหม่ผิดพลาด ซึ่งการผิดพลาดแบบนี้จะต้องแก้ไขโดยเอาอักขระควบคุมเหล่านี้ออกไป สำหรับรายการต่างๆ นั้นจะต้องตรวจว่า แต่ละ field มีรูปแบบที่ถูกต้อง และ อยู่ในค่าที่ถูกต้อง เช่น Timestamps ต้องอยู่ในระยะเวลา ของ Server Log และ รหัสของHTTP Response อยู่ในชุดของค่าที่เป็นไปได้ การตรวจสอบความผิดพลาดของ field และกำจัดด้วย manual

2.4.2 การแปลงให้อยู่ในรูปแบบที่เหมาะสม

แทนที่จะทำการศึกษาต่อไปโดยใช้ Log ที่ผ่านการ Clean นั้น จะต้องแปลงบางข้อมูล Log ให้เป็น Integer ที่เรียงลำดับซึ่งจะทำให้ขนาดของ Log ลดลง โดยการลดจำนวนของ field และขนาด

ของแต่ละ field และยังทำให้ไม่ต้องจัดการเกี่ยวกับ string ที่ยาวไม่คงที่ ในการเขียนโปรแกรมส่วนถัดมาเช่น การแปลงรูปแบบการแสดงวันเวลา ให้อยู่ในรูปตัวเลขในหน่วยของวินาที

ข้อมูลที่ได้มานั้นอาจจะมากเกินไป จึงอาจจะต้องจำกัดจำนวนของ unique Identifiers เพื่อหลีกเลี่ยงการใช้หน่วยความจำ และ การคำนวณที่ซับซ้อนในการทำงานถัดไป เนื่องจากมี Resource ที่ไม่ได้ถูกใช้งานบ่อยตาม Zipf's Law ดังนั้นจึงควรสนใจแต่ Resource ที่ใช้งานบ่อยอย่างเดียว ในทำนองเดียวกัน Request ส่วนใหญ่มาจาก Client กลุ่มหนึ่ง จึงทำให้จำกัดความสนใจต่อกลุ่ม Client นี้ ในการศึกษา Server Prediction จะพัฒนาโดยให้ Server Response Message นั้นมี Hint ของการใช้งานในอนาคตซึ่งมีความซับซ้อนในการคำนวณเพื่อสร้าง Hint ที่ถูกต้องสำหรับ Resource ซึ่งเป็น การดีที่ไม่สร้าง Hint ให้ Piggyback Hint บนการ Response ของ Resource ที่ไม่ได้ถูกใช้งานบ่อยนัก เพื่อเป็นการลดการใช้หน่วยความจำในการคำนวณของการทำนายนั้นจะกำหนดค่า Identifier เดียวกัน ให้ทุก Resource ที่การใช้งานต่ำกว่า Popularity Threshold

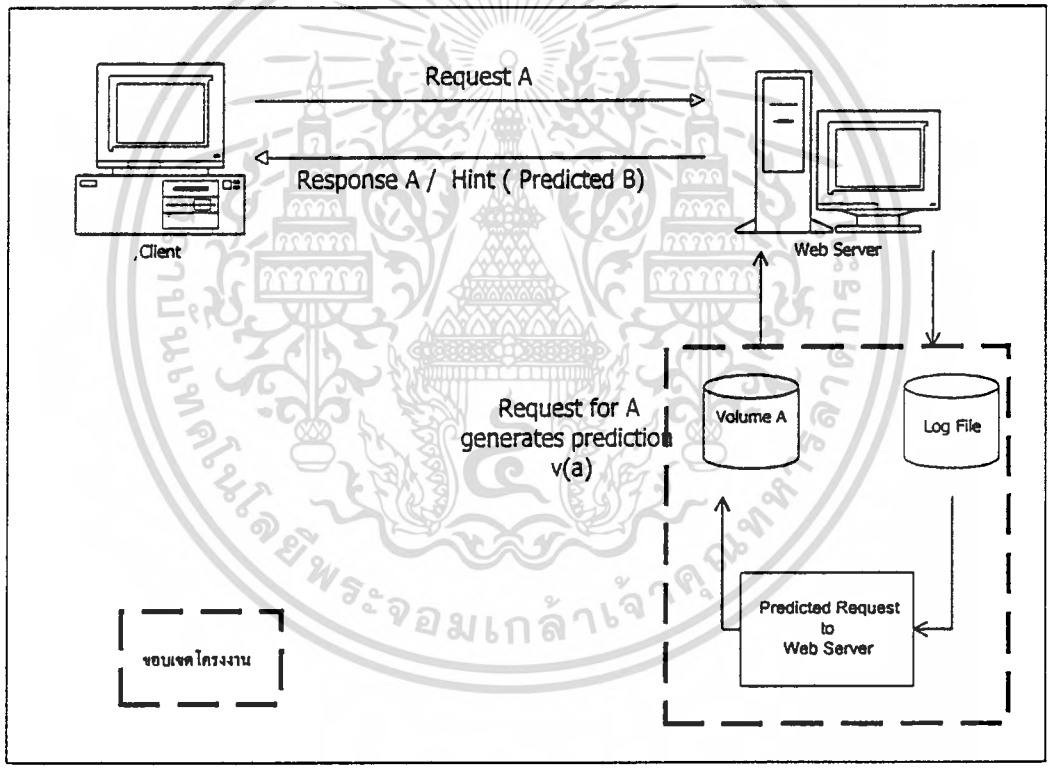
2.4.3 การเรียงลำดับกลุ่มข้อมูล

หลังจากการแปลงรูปแบบและคัดเลือกข้อมูลของ Log ที่ผ่านการ Clean แล้วจะทำงาน เพื่อการวัดค่าต่าง ๆ ในการทำการหารูปแบบการเข้าถึงนั้น เราจะทำรายการของ Log ทั้งหมดตามแต่ละ Client แทนที่จะทำงานตามลำดับของเวลา แล้วคำนวณค่าสถิติต่าง ๆ ตามแต่ละ Client โดยลำดับแรกเรียง Log ตาม Client Identifier ซึ่งทำให้เราพิจารณา Client หนึ่งๆ ที่เวลาใดๆ ได้และเราจะต้องจัดเรียง Client เดียวกัน ตาม TimeStamp จะอยู่ในลำดับที่ถูกต้อง ดังนั้นจึงจะคำนวณค่าสถิติทั้งหมดหลังจากที่ได้จัดเรียงลำดับการ Request ของ Client ให้เหมาะสม

บทที่ 3
การออกแบบโปรแกรมและฐานข้อมูล

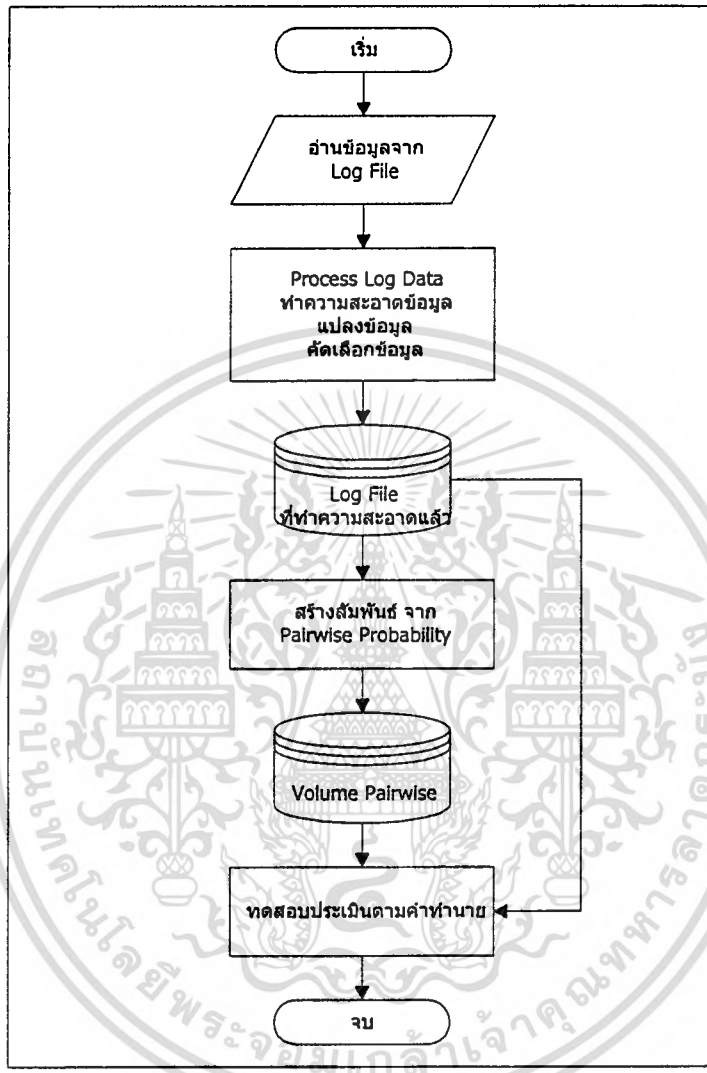
3.1 การออกแบบโปรแกรม

จากแนวทางการศึกษาคือ พัฒนาโปรแกรมหาความสัมพันธ์ แล้ววัดผลการทำนาย ซึ่งจะเห็นว่างานที่เกี่ยวข้องในการพัฒนาแสดงได้ดังรูปข้างล่างนี้



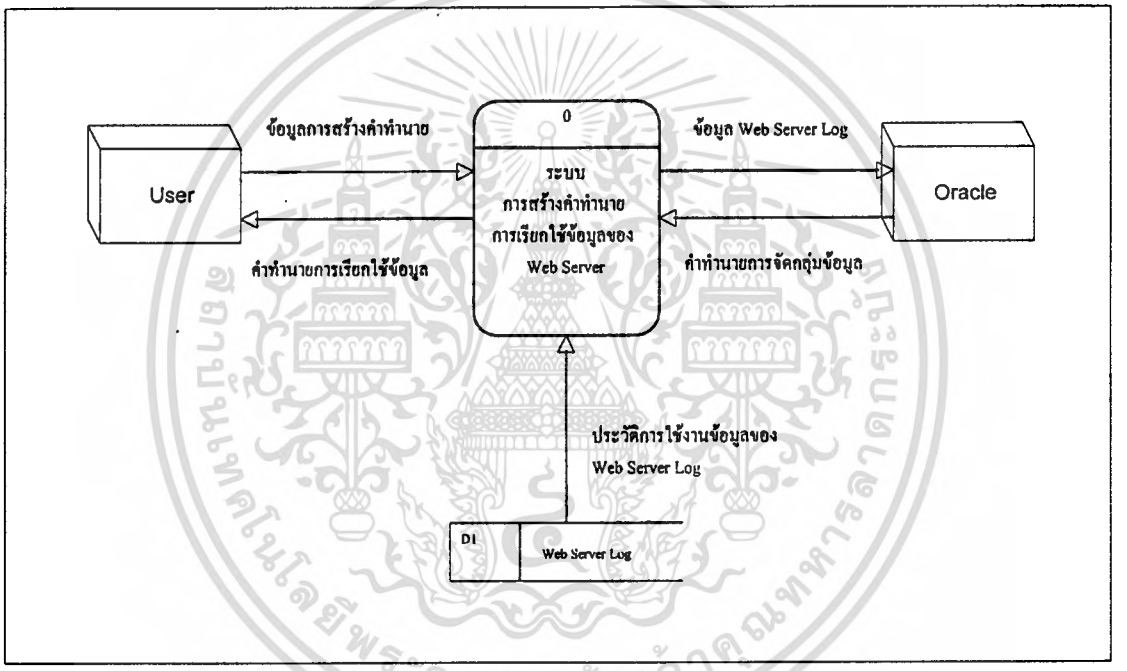
รูปที่ 3.1 ขอบเขตของพัฒนาการสร้างคำทำนายการเรียกใช้ข้อมูลของ Web Server

โดยการพัฒนาจะแบ่งเป็นสองส่วนคือ ส่วนแรกเป็นการวิเคราะห์การจัดกลุ่มข้อมูลเพื่อสร้างคำทำนายการเรียกใช้ข้อมูลโดยสร้างจาก Log File แล้วได้ผลลัพธ์เป็นความสัมพันธ์ของข้อมูลเก็บไว้ใน Volume Pairwise ซึ่งเก็บเป็นตารางฐานข้อมูล ส่วนที่สองคือเป็นการทดสอบประเมินคำทำนายที่ได้ว่ามีประสิทธิภาพเพียงใด โดยอาศัยข้อมูลจาก Log File และเปรียบเทียบคำทำนายจาก Volume Pairwise ว่าคำทำนายมีความถูกต้องเพียงใด โดยแสดงได้ดังรูปที่ 3.2



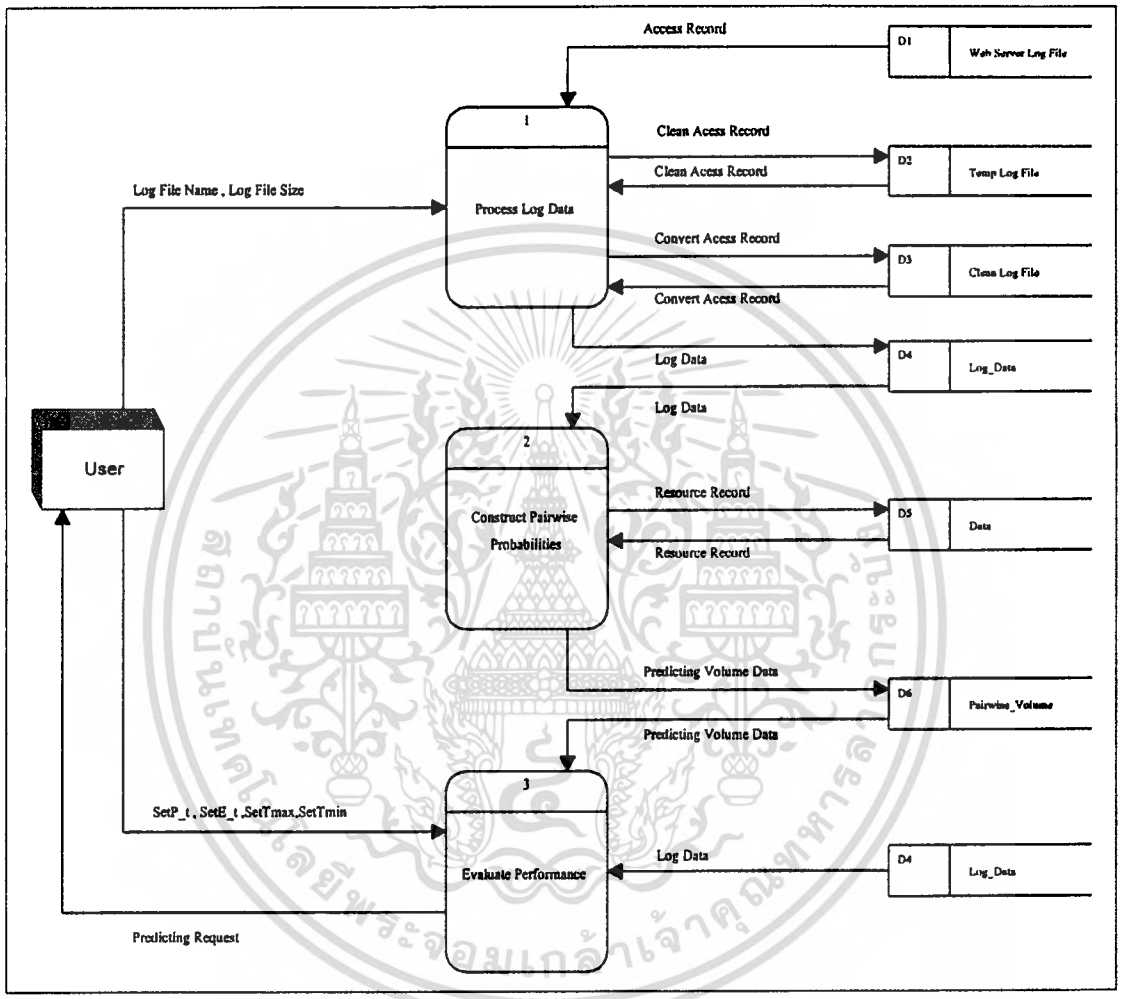
รูปที่ 3.2 ภาพรวมของพัฒนาการสร้างคำทำนายการเรียกใช้ข้อมูลของ Web Server

จากการศึกษาวิเคราะห์การจัดกลุ่มข้อมูลเพื่อสร้างคำทำนายการเรียกใช้ข้อมูลจากเว็ป ได้ออกแบบโปรแกรมให้การทำงานในโปรแกรมออกเป็น 2 ส่วน คือ ส่วนที่ใช้สำหรับเตรียมข้อมูล Web Server Log ที่จะนำมาใช้เป็นข้อมูลอินพุตสำหรับการจัดกลุ่มข้อมูล โดยข้อมูลส่วนนี้ที่ผ่านการจัดเตรียมเรียบร้อยนี้จะถูกเก็บไว้ในฐานข้อมูล เพื่อใช้สำหรับส่วนถัดไป ซึ่งคือส่วนการทำการจัดกลุ่มข้อมูลที่น่าจะถูกเรียกใช้ถัดไปให้อยู่ในกลุ่มเดียวกันโดยใช้การจัดกลุ่มแบบ Pairwise Probabilites โดยขั้นตอนในการทำงานที่ได้ออกแบบไว้นี้สามารถเขียนเป็น context diagram และ data flow diagram ได้ดังแสดงในรูปที่ 3.3 และ 3.4



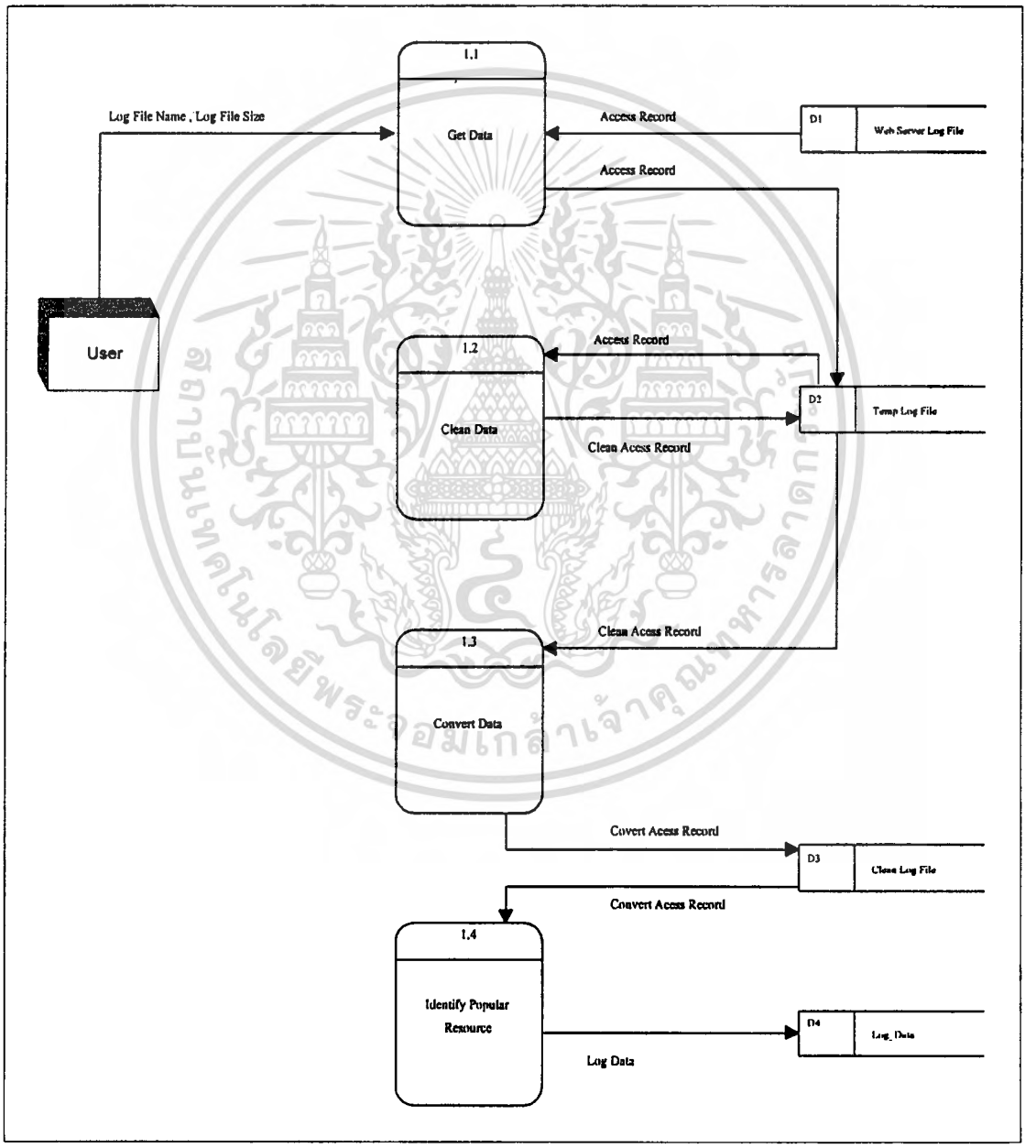
รูปที่ 3.3 Context Diagram ของการพัฒนาระบบการสร้างคำทำนายการเรียกใช้ข้อมูลของ Web Server

จากรูปนำประวัติการเรียกใช้งานข้อมูลจาก Web Server Log File มาใช้เป็นข้อมูลเพื่อหาความสัมพันธ์ของกลุ่มข้อมูลใน Web Server โดยจะต้องผ่านขบวนการทำงานต่างดังรูป 3.4 ข้างล่าง



รูปที่ 3.4 Data Flow Diagram Level 1 ของระบบการสร้างคำทำนายการเรียกใช้ข้อมูลของ Web Server

จากรูปที่ 3.4 งานย่อยที่แตกออกมาประกอบด้วย Process ต่าง 3 ส่วน ในโปรแกรมส่วนที่เป็น Process หมายเลข 1 เป็นงานย่อยของการทำการจัดเตรียมข้อมูลก่อนการเข้าไปสู่กระบวนการจัดกลุ่มข้อมูลตามวิธี Pairwise Probability ของ Process หมายเลข 2 และส่วนสุดท้าย Process หมายเลข 3 เป็นส่วนการประเมินผลลัพธ์การทดสอบโปรแกรม



รูปที่ 3.5 Data Flow Diagram Level 2 ของ Process 1.0 Process Log Data

Process ที่ 1.1 และ 1.2 เป็นการเลือกนำข้อมูลช่วงหนึ่งจาก Log File มาทำความสะอาด โดยจัดการข้อมูลที่มีความซ้ำซ้อนกัน และข้อมูลที่ไม่มีประโยชน์ทิ้ง แล้ว Process ที่ 1.3 แปลงทรัพยากรและเวลาให้อยู่ในรูปตัวเลขที่เหมาะสม พร้อมทั้งเลือกข้อมูลจำนวนหนึ่งซึ่งมีแนวโน้มถูกเรียกใช้งานบ่อยจาก Process 1.4 ซึ่ง Process 1.3 และ 1.4 นี้ทำเพื่อลดขั้นตอนของเวลาและหน่วยความจำที่ใช้ในการทำงานของ Process 2.0 โดยผลลัพธ์ที่ต้องการจากถูกแปลงจาก Log File ลงไปเก็บใน Oracle Database เพื่อให้ Process 2.0 สะดวกในการ Query ข้อมูลตามเงื่อนไขที่ต้องการ

Process 2.0 เป็นการจัดกลุ่มข้อมูลที่มีความน่าจะเป็นว่าจะถูกเรียกใช้พร้อมกันให้อยู่ด้วยกันตามอัลกอริทึมที่ศึกษามา โดยในที่นี้เลือกใช้ Pairwise Probability ซึ่งจะใช้เวลาในการคำนวณน้อยกว่าแบบ Greedy

Process 3.0 เป็นการนำผลที่ได้มาทดสอบเพื่อคำนวณหาค่า Recall , Precision และ Hint Size เพื่อจะได้ประเมินผลความถูกต้องของการจัดกลุ่มนี้

3.2 การออกแบบฐานข้อมูล

ฐานข้อมูลของระบบงานนี้ประกอบด้วยตารางหลักๆ 4 ตารางได้แก่ ตาราง “Log_Data” ซึ่งเป็นที่เก็บข้อมูลการเรียกใช้งานของผู้ใช้ต่อ Web Server ตาราง“Data” เพื่อเก็บรายการทรัพยากรทั้งหมดที่ถูกเรียกใช้งาน สุดท้ายสองตาราง “Pairwise_volume” และ “TempPairwise_volume” นั้นได้จากการสร้างทำงานตามอัลกอริทึม Pairwise โดยเก็บผลลัพธ์เก็บไว้ที่ “Pairwise_volume”

รายละเอียดและความสัมพันธ์ของแต่ละฐานข้อมูลมีดังนี้

1. **Log_Data** ใช้เก็บข้อมูลการเรียกใช้งานต่อ Web Server ซึ่งเป็นข้อมูลที่ผ่านการทำความสะอาดแล้ว โดยมีรายละเอียดดังนี้

Field	Type	Description
ID	Number	หมายเลข ID ของรายการในตารางนี้
IP	VarChar2(20)	IP Address ซึ่งระบุถึง User ที่เรียกใช้ข้อมูลของ Web Server
Sec	Number	วินาทีที่เรียกใช้ข้อมูลของ Web Server
Data1	VarChar2(500)	ข้อมูลทรัพยากร(ไฟล์ที่ถูกเรียกใช้งาน) ซึ่งผ่านการแปลงให้อยู่ในรูปตัวเลข

2. **Data** ใช้เก็บข้อมูลรายการทรัพยากรทั้งหมด (ไฟล์ที่ถูกเรียกใช้งาน) โดยมีรายละเอียดดังนี้

ID_New	Number	หมายเลข ID ของรายการในตารางนี้

ID_Old	Number	หมายเลข ID ที่อ้างถึงรายการในตาราง Log_Data
Data1	VarChar2(500)	ข้อมูลทรัพยากร(ไฟล์ที่ถูกเรียกใช้งาน)
Count_data1	Number	นับจำนวนของไฟล์นั้นๆ ตามจำนวนของการเรียกใช้งาน

3. **TempPairwise_Volume** ใช้เก็บข้อมูลซึ่งได้หาความสัมพันธ์ตาม Pairwise Probability โดยมีรายละเอียดดังนี้

ID	Number	หมายเลข ID ของรายการในตารางนี้
B	VarChar2(500)	ข้อมูลไฟล์ที่ถูกเรียกใช้งานถัดจาก A
A	VarChar2(500)	ข้อมูลไฟล์ที่ถูกเรียกใช้งาน
C_BA	Number	จำนวนครั้งของการเรียกงานไฟล์ A แล้วตามด้วย B
E_BA	Number	จำนวนครั้งของการเรียกงานไฟล์ A แล้วตามด้วย B เป็นไฟล์แรก

4. **Pairwise_Volume** ใช้เก็บผลลัพธ์ทั้งหมดที่เกี่ยวข้องที่จะต้องนำไปใช้ในการ Predicting Request โดยมีรายละเอียดดังนี้

ID	Number	หมายเลข ID ของรายการในตารางนี้
B	VarChar2(500)	ข้อมูลไฟล์ที่ถูกเรียกใช้งานถัดจาก A
A	VarChar2(500)	ข้อมูลไฟล์ที่ถูกเรียกใช้งาน
P_BA	Number	Probability ของการเรียกงานไฟล์ A แล้วตามด้วย B
E_BA	Number	Effectiveness ของการเรียกงานไฟล์ A แล้วตามด้วย B เป็นไฟล์แรก

บทที่ 4
การพัฒนาโปรแกรม

4.1 หลักการทำงานของโปรแกรม

หลังจากที่ได้ออกแบบโปรแกรมเรียบร้อยแล้ว จึงพิจารณาส่วนการทำงานของโปรแกรมออกเป็น 2 ส่วนหลักๆ คือ ส่วนของการจัดการข้อมูล หรือ การเตรียมข้อมูล และ ส่วนของการจัดกลุ่มข้อมูลตามอัลกอริทึม Pairwise Probability ในการพัฒนาโปรแกรมตามการศึกษานี้ได้ใช้ Borland C++ และ Oracle8I ในการพัฒนาโปรแกรม

4.1.1 ส่วนการจัดการข้อมูล

ในส่วนแรกจะเป็นส่วนที่นำเอาข้อมูลดิบดังแสดงในรูปที่ 4.1 ได้จากไฟล์ Log File ของ Web Server โดยเลือกมาตามขนาดที่เหมาะสมเนื่องจากข้อมูลมีขนาดใหญ่มาก แล้วทำการแปลงข้อมูลให้อยู่ในรูปแบบที่ง่ายต่อการคำนวณ และ ใช้หน่วยความจำน้อยลงในการทำงานส่วนถัดไป

```
203.151.63.137 - - [23/Jun/2000:18:11:46 +0700] "GET /~s391/ALP-SILVER-A.GIF HTTP/1.0" 200 4815
161.246.11.242 - - [23/Jun/2000:18:11:49 +0700] "GET /board/sub_khaesad/subj5116.htm HTTP/1.1" 200 1537
203.149.1.60 - - [23/Jun/2000:18:11:51 +0700] "GET /bksnoscene.jpg HTTP/1.1" 404 291
194.131.26.2 - - [23/Jun/2000:18:11:52 +0700] "GET /~s013/Link/16.html HTTP/1.1" 200 707
203.151.63.137 - - [23/Jun/2000:18:11:52 +0700] "GET /~s391/ HTTP/1.0" 200 1074
203.146.60.194 - - [23/Jun/2000:18:11:52 +0700] "-" 408 –
161.246.11.242 - - [23/Jun/2000:18:12:03 +0700] "GET /board/sub_khaesad/subj5115.htm HTTP/1.1" 200 1647
203.151.63.137 - - [23/Jun/2000:18:12:10 +0700] "GET /~s391/header.html HTTP/1.0" 200 2011
203.154.33.180 - - [23/Jun/2000:18:12:11 +0700] "GET /photo HTTP/1.0" 301 304
203.154.33.180 - - [23/Jun/2000:18:12:11 +0700] "GET /photo/ HTTP/1.0" 200 214
203.149.17.20 - - [23/Jun/2000:18:12:11 +0700] "GET /board/reload_khaesad.html HTTP/1.0" 304 -
203.151.63.137 - - [23/Jun/2000:18:12:11 +0700] "GET /~s391/menu.html HTTP/1.0" 200 2232
203.151.63.137 - - [23/Jun/2000:18:12:13 +0700] "GET /~s4391/main.html HTTP/1.0" 200 2563
203.151.63.137 - - [23/Jun/2000:18:12:13 +0700] "GET /~s4391/newsov.gif HTTP/1.0" 200 1820
203.151.63.137 - - [23/Jun/2000:18:12:13 +0700] "GET /~s4391/main.jpg HTTP/1.1" 200 7612
```

รูปที่ 4.1 ตัวอย่างข้อมูลจาก Web Server Log File

- การทำความสะอาดข้อมูล (Cleaning Process) ทำการอ่านข้อมูลตาม Format ของ Log File ในขั้นแรกจะทำการลบรายการที่ไม่มีประโยชน์ออกไปซึ่งรายการที่พบเช่น

```
203.146.60.194 - - [31/Jan/2001:15:06:21 +0700] "-" 408 –
```

รูปที่ 4.2 แสดงถึงรายการที่ไม่มีประโยชน์

จะเห็นว่าข้อมูลนี้แสดงถึง "-" ซึ่งไม่ได้แสดงถึงทรัพยากรที่ถูกเรียกใช้งานของ Web Server

```
202.22.33.253^203130082^/sportsday/2000
202.22.33.253^203130082^/sportsday/2000/
```

รูปที่ 4.3 แสดงถึงรายการที่ซ้ำซ้อนกัน

จะเห็นว่าจากการทำงานของ Web Server อาจจะสร้างข้อมูลสองข้อมูล มีความซ้ำซ้อนกัน ดังนั้นจึงต้องลบรายการที่ซ้ำกันออกไป

```
161.246.11.242 - - [23/Jun/2000:18:13:24 +0700] "GET /electronics/wwwboard3/Data/1495.htm HTTP/1.1" 200 2692
161.246.11.242 - - [23/Jun/2000:18:13:24 +0700] "GET /electronics/wwwboard3/bob2.gif HTTP/1.1" 304 -
161.246.11.242 - - [23/Jun/2000:18:13:24 +0700] "GET /electronics/wwwboard3/line.gif HTTP/1.1" 304 -
161.246.11.242 - - [23/Jun/2000:18:13:24 +0700] "GET /electronics/wwwboard3/bub2.gif HTTP/1.1" 304 -
161.246.11.242 - - [23/Jun/2000:18:13:24 +0700] "GET /electronics/wwwboard3/aster.gif HTTP/1.1" 200 121
```

รูปที่ 4.4 แสดงถึงรายการที่ไม่ได้เกิดจากการเรียกจากผู้ใช้

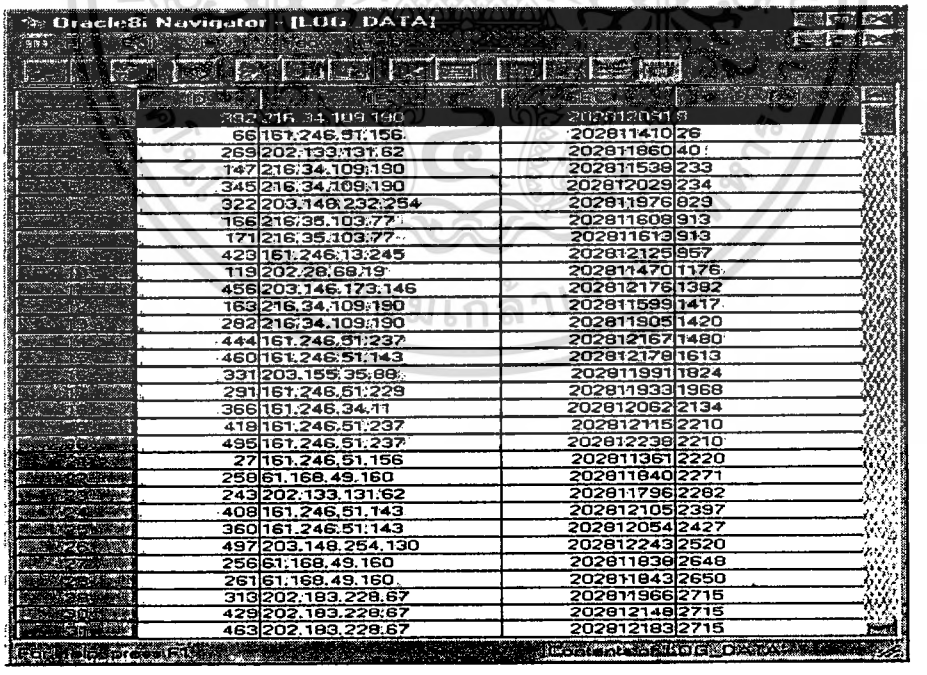
จะเห็นว่าในการเรียกใช้งานเพจหนึ่งๆ จะทำให้เกิดรายการหลายรายการ เนื่องจากต้องมีการดาวน์โหลดรูปกราฟฟิกต่างๆ เข้ามาเพิ่มเติมที่เพจของผู้ใช้ดังนั้นจึงต้องกำจัดรายการส่วนเกินดังกล่าวออกไป

จากการพิจารณาถึงอัลกอริทึมจะพบว่าข้อมูลที่จำเป็นต้องใช้ 3 ค่า คือ IP Address (user), เวลา และ ไฟล์ ดังนั้นจึงทำการลบข้อมูลส่วนที่นอกเหนือจากนี้ทิ้งไปโดยลักษณะของ Log File ที่ได้จะแสดงดังรูปที่ 4.5

```
161.246.11.242^23/Jun/2000:18:11:49^/board/sub_khaesad/subj5116.htm
194.131.26.2^23/Jun/2000:18:11:52^/~s013/Link/16.html
203.151.63.137^ 23/Jun/2000:18:11:52^/~s391/
161.246.11.242^23/Jun/2000:18:12:03 ^/board/sub_khaesad/subj5115.htm
203.151.63.137 ^23/Jun/2000:18:12:10^/~s391/header.html
203.154.33.180 ^23/Jun/2000:18:12:11 ^/photo/
203.149.17.20 ^23/Jun/2000:18:12:11^/board/reload_khaesad.html
203.151.63.137 ^23/Jun/2000:18:12:11^/~s4391/menu.html
203.151.63.137 ^23/Jun/2000:18:12:13^/~s4391/main.html
```

รูปที่ 4.5 แสดงถึงตัวอย่าง Log File ที่ผ่านการทำความสะอาดแล้ว

- การแปลงรูปแบบข้อมูล (Convert Data) โดยนำข้อมูลที่ได้ทำความสะอาดแล้วดังกล่าวมาแปลงการแสดงผลเวลาให้อยู่ในรูปวินาที และ แปลงข้อมูลไฟล์ให้อยู่ในรูปตัวเลขเพื่อลดเวลาในการคำนวณเปรียบเทียบและลดการใช้หน่วยความจำในการทำงานถัดไป โดยผลที่ได้จะนำไปเก็บไว้ในตาราง Log_Data ดังแสดงดังรูป 4.6 นี้

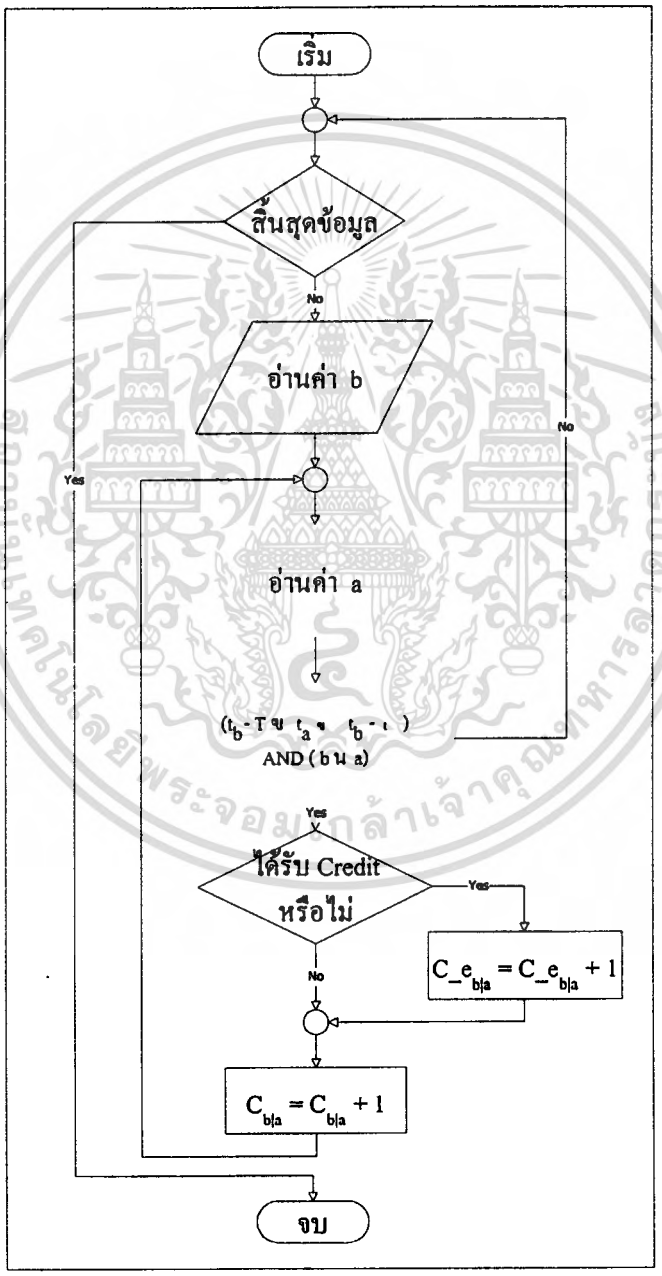


392	216.34.109.190	202812031	8
66	161.246.51.156	202811410	26
269	202.133.131.62	202811860	40
147	216.34.109.190	202811539	233
345	216.34.109.190	202812029	234
322	203.148.232.254	202811976	829
166	216.35.103.77	202811608	913
171	216.35.103.77	202811613	913
429	161.246.13.245	202812125	957
119	202.28.68.19	202811470	1176
456	203.146.173.146	202812176	1382
163	216.34.109.190	202811599	1417
282	216.34.109.190	202811905	1420
444	161.246.51.237	202812167	1480
460	161.246.51.143	202812178	1613
331	203.155.35.66	202811991	1824
291	161.246.51.229	202811933	1968
366	161.246.34.11	202812062	2134
418	161.246.51.237	202812115	2210
495	161.246.51.237	202812238	2210
27	161.246.51.156	202811361	2220
259	61.168.49.160	202811840	2271
243	202.133.131.62	202811796	2282
408	161.246.51.143	202812105	2397
360	161.246.51.143	202812054	2427
497	203.148.254.130	202812243	2520
256	61.168.49.160	202811839	2648
261	61.168.49.160	202811843	2650
313	202.183.228.67	202811966	2715
429	202.183.228.67	202812148	2715
463	202.183.228.67	202812183	2715

รูปที่ 4.6 แสดงตัวอย่างข้อมูลตาราง Log_Data

4.1.2 ส่วนการจัดกลุ่มข้อมูลตามอัลกอริทึม Pairwise Probability

หลังจากที่ได้ข้อมูลจากส่วนแรกแล้วขั้นตอนต่อไปเป็นการนำเอาข้อมูลดังกล่าวนั้นมาจัดกลุ่มข้อมูลที่มีความน่าจะเป็นว่าจะถูกเรียกใช้ในลำดับถัดกันให้อยู่ด้วยกันตามวิธีของ Pairwise Probability ดังที่ได้อธิบายมาแล้วในบทที่ 2 โดยจะแสดงตัวอย่างการนับ Counter และ ตัวแปรต่างๆ ที่เกี่ยวข้องดังนี้



รูปที่ 4.7 Flow Chart การนับค่า Counter ต่างๆ

- ตัวอย่างการคำนวณหา $P_{b|a}$

Access : a_1 , b_1 , c , ... , b_2 , d , ... a_2, d , ... a_3 , d , a_4 , b_3 ,a_5 , e, b_4 , ... a_6 ,f

ในขณะนี้จะอธิบายเฉพาะความสัมพันธ์ a กับ b และพิจารณาในช่วงเวลาที่เหมาะสม

ขั้นตอนที่ 1 นับจำนวนของแต่ละข้อมูลจะได้ว่า $C_a = 6$ **และ** $C_b = 4$

ขั้นตอนที่ 2 นับจำนวนความสัมพันธ์ตามเวลาที่กำหนด โดยในโปรแกรมเริ่มพิจารณาการนับมาจากส่วนท้ายเรียงตามเวลาจาก b_4 ,b_3 , b_2 , b_1 **คือ**

- b_4 ค้นพบ a_5 ดังนั้นค่า $C_{b4|a} = 1$ และ มาหยุดการพิจารณา b_4 เมื่อมาเจอ b_3
- b_3 ค้นพบ a_4 และ a_3 ดังนั้นค่า $C_{b3|a} = 2$ และ มาหยุดการพิจารณา b_3 เนื่องจากเลยช่วงเวลาที่กำหนด
- b_2 ในช่วงเวลาที่กำหนดไม่พบ a ใดๆ
- b_1 ค้นพบ a_1 ดังนั้นค่า $C_{b1|a} = 1$

ขั้นตอนการคำนวณผลรวม $C_{b|a} = 4$ จาก คู่ (a_5, b_4) , (a_4, b_3) , (a_3, b_3) และ (a_1 , b_1)

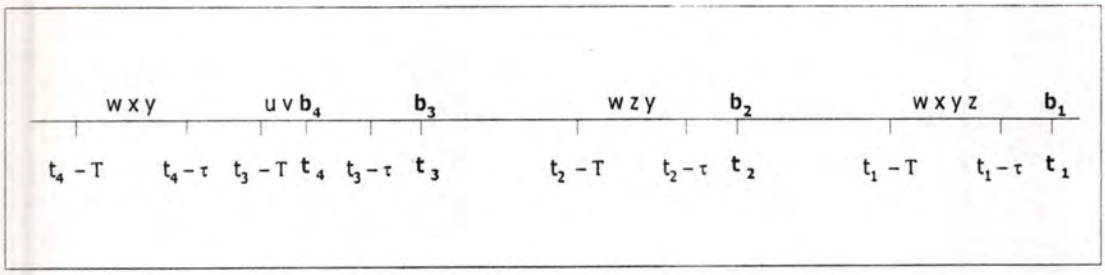
$$P_{b|a} = C_{b|a} / C_a = 4/6$$

- ตัวอย่างการคำนวณหา $E_{b|a}$

โดยในการโปรแกรมนี้ให้เครดิตการทำนายกับตัวที่อยู่หลังสุดคือตัวที่อยู่ใกล้ข้อมูล b มากที่สุด ดังนั้น $E_{b|a}$ ที่ได้จะได้มาจากคู่ (a_4, b_3) และ (a_1 , b_1) ส่วนคู่อื่นที่เหลือนั้น a ไม่ได้รับเครดิตในการสร้างคำทำนาย b เช่น คู่ (a_5, b_4) นั้น จะได้ค่าหลังสุดก่อน b_4 คือ e หมายถึง e ได้รับเครดิตในการทำนาย

ดังนั้นผลการคำนวณคือ $E_{b|a} = (C_{b|a}$: ตัวที่ได้เครดิตการทำนาย) / $C_{b|a}$

$= 2/4$



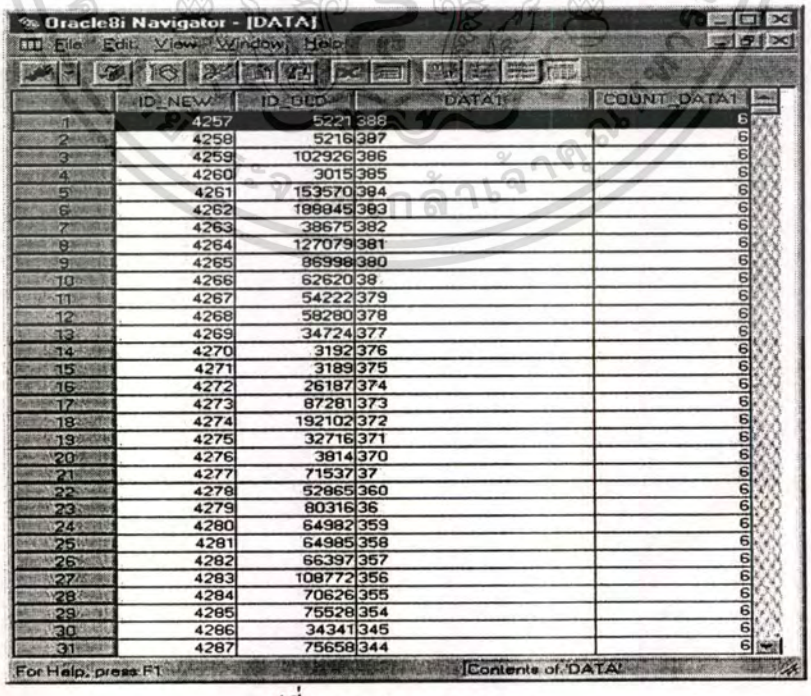
รูปที่ 4.8 ตัวอย่างการนับค่า Counter ต่างๆ

พิจารณาผลรวมของ Counter (C) ต่างๆ ที่เวลาต่างๆ (คำนวณย้อนมาจากหลัง)

- $t_1 : C_{b|w} = 1 , C_{b|x} = 1 , C_{b|y} = 1 , C_{b|z} = 1 , C\_e_{b|z} = 1$
- $t_2 : C_{b|w} = 2 , C_{b|x} = 1 , C_{b|y} = 2 , C_{b|z} = 2 , C\_e_{b|z} = 1 , C\_e_{b|y} = 1$
- t_3 : C ทุกค่าคงเดิมตาม t_2
- $t_4 : C_{b|w} = 3 , C_{b|x} = 2 , C_{b|y} = 3 , C_{b|z} = 2 , C\_e_{b|z} = 1 , C\_e_{b|y} = 2$

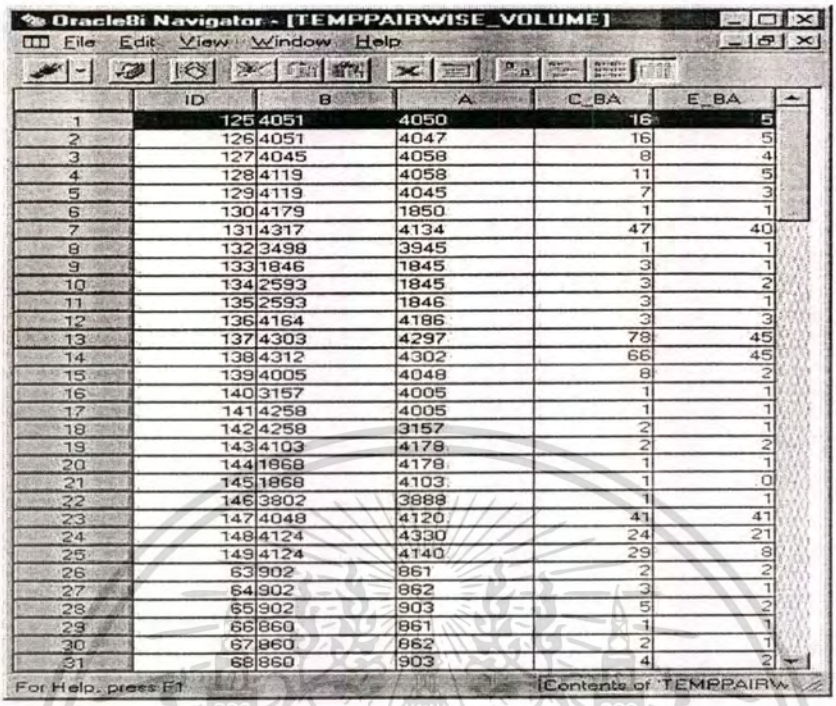
สรุป : $P_{b|w} = 3/3 , P_{b|x} = 2/3 , P_{b|y} = 3/3 , P_{b|z} = 2/3 , E_{b|z} = 1/2 , E_{b|y} = 2/3$

โดยนำผลลัพธ์ที่ได้ไปเก็บไว้ในฐานข้อมูลตารางดังรูปข้างล่างต่อไปนี้



	ID_NEW	ID_OLD	DATA1	COUNT_DATA1
1	4257	5221	388	6
2	4258	5216	387	6
3	4259	102926	386	6
4	4260	3015	385	6
5	4261	153570	384	6
6	4262	188045	383	6
7	4263	38675	382	6
8	4264	127079	381	6
9	4265	86998	380	6
10	4266	62620	38	6
11	4267	54222	379	6
12	4268	58280	378	6
13	4269	34724	377	6
14	4270	3192	376	6
15	4271	3189	375	6
16	4272	26187	374	6
17	4273	87281	373	6
18	4274	192102	372	6
19	4275	32716	371	6
20	4276	3814	370	6
21	4277	71537	37	6
22	4278	52865	360	6
23	4279	80316	36	6
24	4280	64982	359	6
25	4281	64985	358	6
26	4282	66397	357	6
27	4283	108772	356	6
28	4284	70626	355	6
29	4285	75528	354	6
30	4286	34341	345	6
31	4287	75658	344	6

รูปที่ 4.9 แสดงตาราง Data



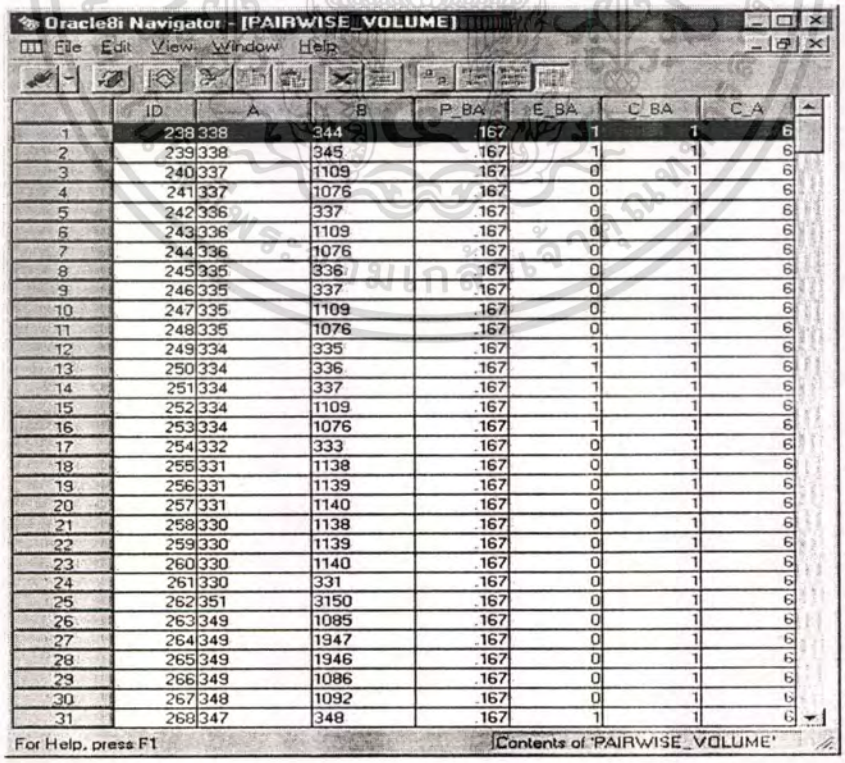
Oracle8i Navigator - [TEMPPAIRWISE_VOLUME]

File Edit View Window Help

	ID	B	A	C_BA	E_BA
1	125	4051	4050	16	5
2	126	4051	4047	16	5
3	127	4045	4058	8	4
4	128	4119	4058	11	5
5	129	4119	4045	7	3
6	130	4179	1850	1	1
7	131	4317	4134	47	40
8	132	3498	3945	1	1
9	133	1846	1845	3	1
10	134	2593	1845	3	2
11	135	2593	1846	3	1
12	136	4164	4186	3	3
13	137	4303	4297	78	45
14	138	4312	4302	66	45
15	139	4005	4048	8	2
16	140	3157	4005	1	1
17	141	4258	4005	1	1
18	142	4258	3157	2	1
19	143	4103	4178	2	2
20	144	1868	4178	1	1
21	145	1868	4103	1	0
22	146	3802	3888	1	1
23	147	4048	4120	41	41
24	148	4124	4330	24	21
25	149	4124	4140	29	8
26	63	902	861	2	2
27	64	902	862	3	1
28	65	902	903	5	2
29	66	860	861	1	1
30	67	860	862	2	1
31	68	860	903	4	2

For Help, press F1 Contents of 'TEMPPAIRW...

รูปที่ 4.10 แสดงตาราง TempPairwise_Volume



Oracle8i Navigator - [PAIRWISE_VOLUME]

File Edit View Window Help

	ID	A	B	P_BA	E_BA	C_BA	C_A
1	238	338	344	.167	1	1	6
2	239	338	345	.167	1	1	6
3	240	337	1109	.167	0	1	6
4	241	337	1076	.167	0	1	6
5	242	336	337	.167	0	1	6
6	243	336	1109	.167	0	1	6
7	244	336	1076	.167	0	1	6
8	245	335	336	.167	0	1	6
9	246	335	337	.167	0	1	6
10	247	335	1109	.167	0	1	6
11	248	335	1076	.167	0	1	6
12	249	334	335	.167	1	1	6
13	250	334	336	.167	1	1	6
14	251	334	337	.167	1	1	6
15	252	334	1109	.167	1	1	6
16	253	334	1076	.167	1	1	6
17	254	332	333	.167	0	1	6
18	255	331	1138	.167	0	1	6
19	256	331	1139	.167	0	1	6
20	257	331	1140	.167	0	1	6
21	258	330	1138	.167	0	1	6
22	259	330	1139	.167	0	1	6
23	260	330	1140	.167	0	1	6
24	261	330	331	.167	0	1	6
25	262	351	3150	.167	0	1	6
26	263	349	1085	.167	0	1	6
27	264	349	1947	.167	0	1	6
28	265	349	1946	.167	0	1	6
29	266	349	1086	.167	0	1	6
30	267	348	1092	.167	0	1	6
31	268	347	348	.167	1	1	6

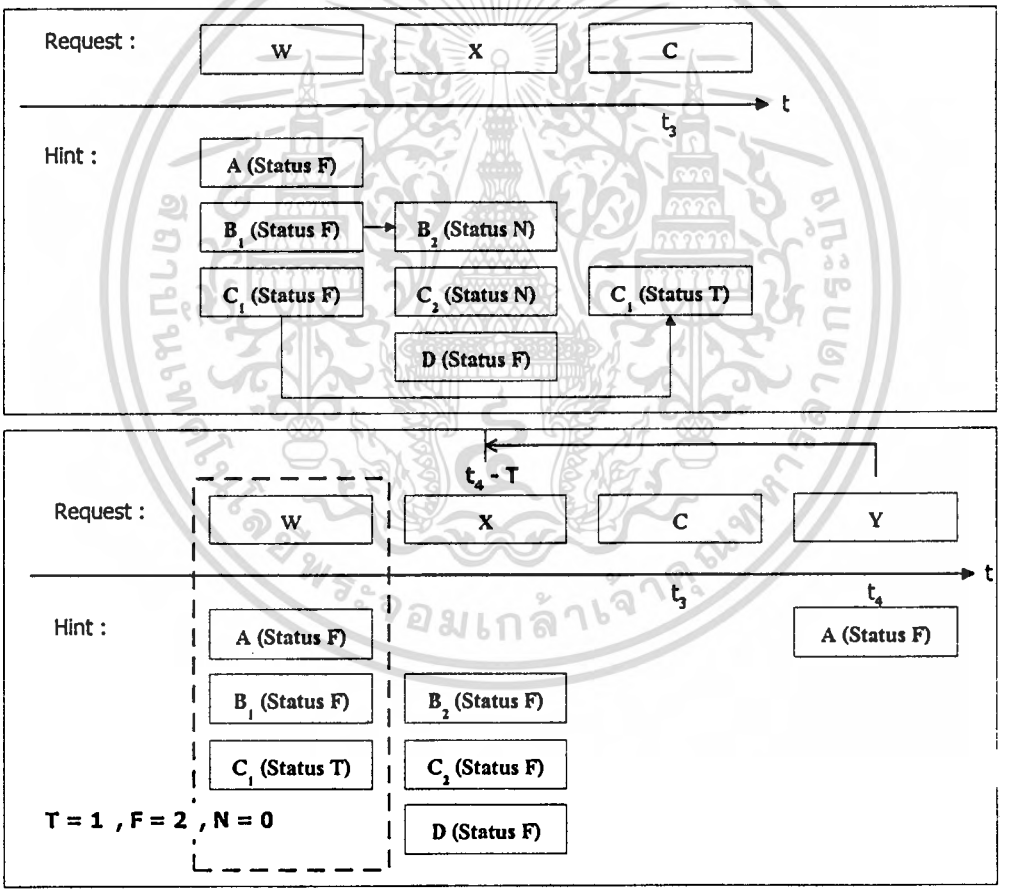
For Help, press F1 Contents of 'PAIRWISE_VOLUME'

รูปที่ 4.11 แสดงตาราง Pairwise_Volume

4.1.3 ส่วนการทดสอบประเมินคำทำนาย

หลังจากสร้าง Volume เก็บความสัมพันธ์ของข้อมูลแล้วขั้นตอนสุดท้ายนี้คือจะทดสอบวัดประสิทธิภาพของคำทำนายตามเกณฑ์ Recall , Precision และ Hint Size จากที่กล่าวมาแล้วก่อนหน้านี้

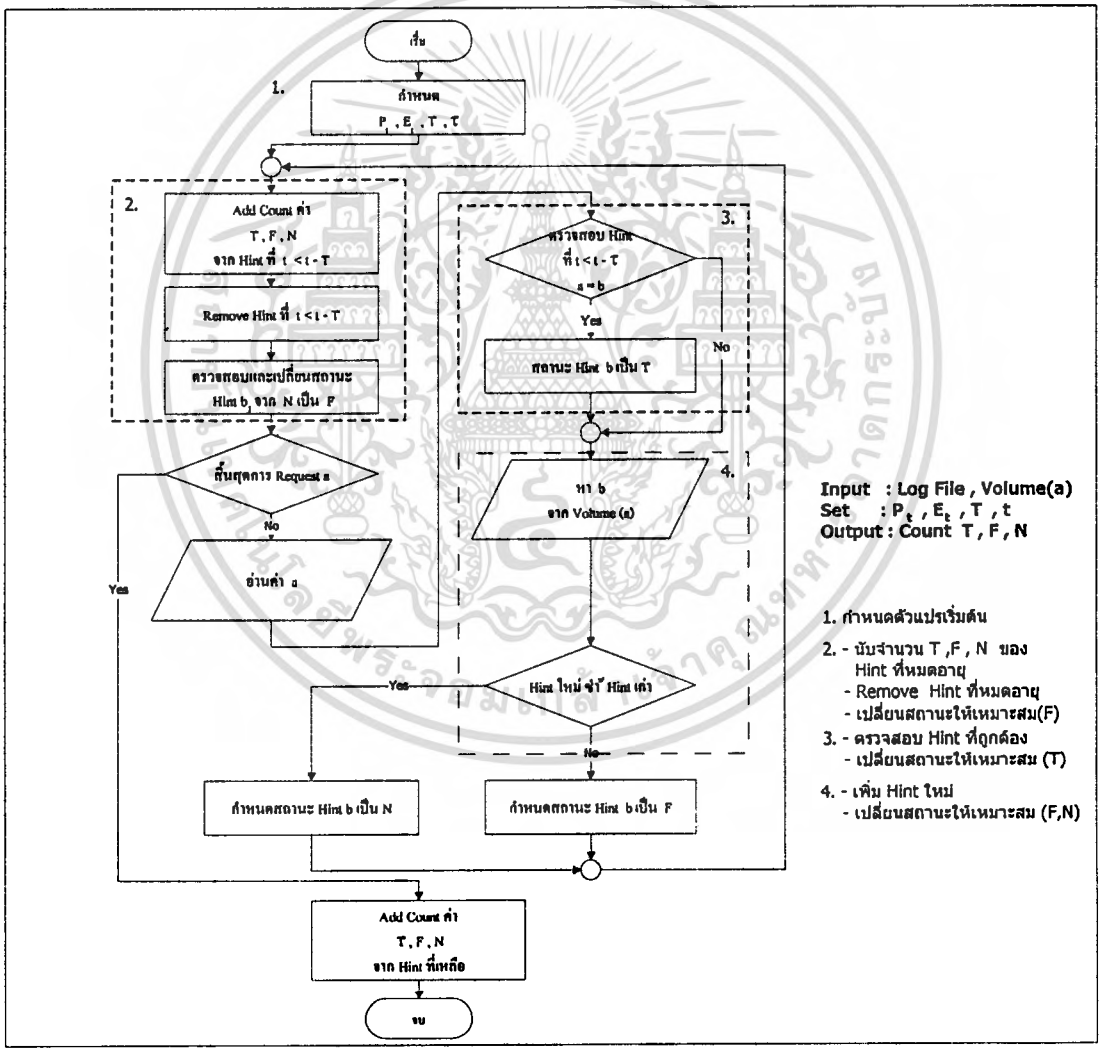
โดยเริ่มต้นคำทำนายที่เกิดขึ้นจะกำหนดให้มีสถานะเป็น F หมายถึงยังไม่ได้มีการเรียกใช้งานจริง เมื่อเวลาผ่านไปหากถูกเรียกใช้งานจะเปลี่ยนสถานะเป็น T และหากพบว่ามีคำทำนายที่เกิดใหม่มีความซ้ำซ้อนกับคำทำนายเก่าจะกำหนดสถานะคำทำนายที่เกิดภายหลังเป็น N ซึ่งคำทำนายที่เกิดขึ้นต่างๆ นั้นจะนำมาพิจารณาในช่วงเวลาหนึ่งที่กำหนดเท่านั้น ซึ่งแสดงได้ดังรูปที่ 4.12



รูปที่ 4.12 ตัวอย่างการเปลี่ยนสถานะคำทำนาย

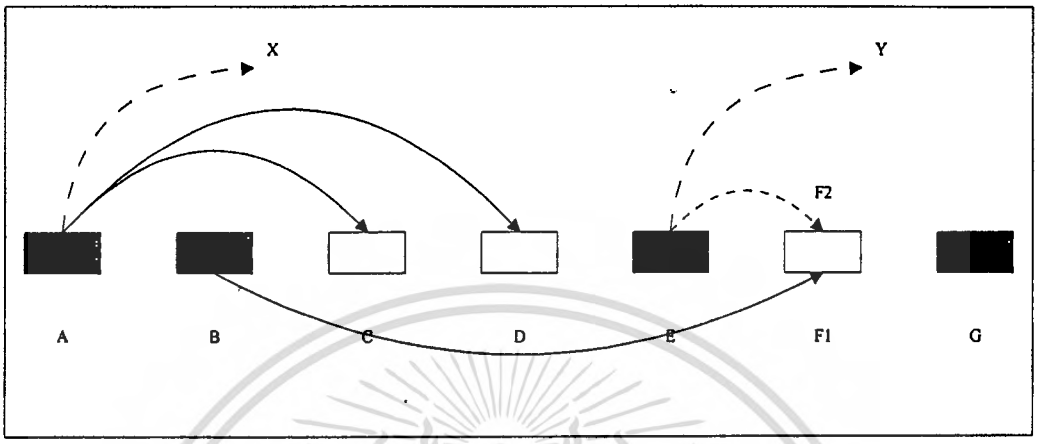
หลังจากการกำหนดสถานะต่างๆแล้ว ก็จะนับจำนวนของสถานะเหล่านั้น เพื่อนำค่าจำนวนนับที่ได้เหล่านั้นมาประเมินประสิทธิผลของคำทำนาย โดยแสดงการทำงานโดยรวมของโปรแกรมได้ดังรูปที่ 4.13 ซึ่งจะเห็นว่าประกอบด้วย 4 ขั้นตอนหลัก คือ

- การกำหนดตัวแปรเริ่มต้น หมายถึง ค่า Threshold และเวลาต่างๆที่จะพิจารณา
- การนับจำนวนของสถานะต่างๆ
- การตรวจสอบหาว่าคำทำนายถูกนำมาใช้จริงหรือไม่
- การเพิ่มคำทำนายตามการเรียกใช้งาน



รูปที่ 4.13 Flow Chart การทดสอบประเมินตามคำทำนาย

- ตัวอย่างการคำนวณประเมินผลเพื่อวัดค่า Recall , Precision และ Hint Size



รูปที่ 4.14 แสดงการทำนายการเรียกใช้ข้อมูล

จากรูปข้างต้นกำหนดว่าคำทำนายและข้อมูลอยู่ในช่วงเวลา [τ, T) วินาที ตามเงื่อนไขที่กำหนด แล้วสามารถอธิบายได้ดังนี้

- ข้อมูล A , B , E , G คือ ข้อมูลที่ถูกเรียกใช้งาน โดยไม่ได้มาจากคำทำนายใดๆ เลย
- ข้อมูล C, D , F1 คือคำทำนายที่ถูกเรียกใช้งาน กำหนดสถานะเป็น Hint_T
- ข้อมูล X ,Y คือ คำทำนายที่ไม่ได้ถูกนำมาใช้จริง กำหนดสถานะเป็น Hint_F
- ข้อมูล F2 คือ คำทำนายที่ซ้ำซ้อน กำหนดสถานะ เป็น Hint_N

จากบทที่ 2 ที่ผ่านมาได้มีการกำหนดนิยามของ Recall , Precison และ Hint Size ไว้แล้วดังนั้น จาก

Recall คือสัดส่วนของข้อมูลที่ถูกเรียกใช้ โดยที่ข้อมูลที่จะถูกเรียกใช้นั้นจะต้องปรากฏอยู่ในคำทำนายในช่วงเวลา [τ, T) วินาทีที่ผ่านมา

Precision คือสัดส่วนของคำทำนาย โดยที่คำทำนายนั้นควรจะถูกใช้งานในช่วงเวลา[τ, T) วินาที ถัดไป

Average Hintsize คือ จำนวนเฉลี่ยของคำทำนายจากในข้อมูลจาก Server ไปยัง Client ดังนั้นจะได้ว่า

Recall = (จำนวนการเรียกใช้ข้อมูลที่มีอยู่ในคำทำนาย) / (จำนวนการเรียกใช้งานข้อมูล)

= (C , D , F { F1 , F2 }) / (C , D , F { F1 , F2 } + A , B , G)

= 3/(3+4)

= (จำนวน Hint_T)/(จำนวน Request)

Precision = (จำนวนคำทำนายที่ถูกใช้งาน) / (จำนวนคำทายทั้งหมด)

= (C , D , F { F1 , F2 }) / (C , D , F { F1 , F2 } + X , Y)

= 3/(3+2)

= (จำนวน Hint_T)/(จำนวน Hint_T + จำนวน Hint_F)

Hint size = (จำนวนคำทำนาย) / (จำนวนการเรียกใช้ข้อมูล)

= (C ,D ,F1 + X ,Y ,F2) / (A , B ,C ,D ,E ,F ,G)

= (3+3) / (3+4)

= (จำนวน Hint_T + จำนวน Hint_F + Hint_N) / (จำนวน Request)

4.2 การทดสอบโปรแกรม

ในการทดสอบนี้เริ่มแรกหาความสัมพันธ์จาก Server Log ของ www.kmitl.ac.th โดยข้อมูลที่นำมาศึกษามีลักษณะดังนี้

- มีรูปแบบการบันทึกเป็น Apache's ECLF
- คัดมา 2,000,000 Request แล้วนำไปผ่านการ Cleaning ได้เป็นข้อมูลจำนวน 462,317 Request
- ทำการทดสอบที่ 70,000 Request

Server Log	Number Requests	Unique Client	Unique Resource
kmitl.ac.th	70,000	1694	3538

ตารางที่ 4.1 ลักษณะของ Server Log

4.2.1 ผลการทดสอบ

ผลที่แสดงต่อไปนี้คือเป็นการทดสอบหลังจากหาความสัมพันธ์ได้แล้วก็นำความสัมพันธ์เหล่านั้นมาสร้างคำทำนายเพื่อดูประสิทธิภาพในการทำนายที่เงื่อนไขต่างๆดังแสดงตามแต่ละตาราง

Probability Threshold	Hint ที่ถูกต้อง	Hint ที่ผิด	Hint ที่ซ้ำซ้อน	จำนวน Request ที่สร้าง Hint	Recall	Precision	Average Hintsize
0.2	19944	56250	13426	31911	0.2849	0.2618	1.2803
0.3	17580	31746	9630	27592	0.2511	0.3564	0.8422
0.4	16362	23450	8455	24965	0.2337	0.4110	0.6895
0.5	15787	19473	7029	23896	0.2255	0.4477	0.6041
0.6	14491	14632	5948	21799	0.2070	0.4976	0.5010
0.7	12976	10375	4280	17339	0.1854	0.5557	0.3947
0.75	6786	5998	3711	10011	0.0969	0.5308	0.2356
0.8	6084	4956	2904	8684	0.0869	0.5511	0.1992
0.85	5679	4304	2335	7871	0.0811	0.5689	0.1760
0.9	2598	2112	987	3448	0.0371	0.5516	0.0814
0.95	1624	1135	566	1773	0.0232	0.5886	0.0475
1	1496	948	490	1512	0.0214	0.6121	0.0419

ตารางที่ 4.2 ผลการทดสอบที่ $T = 60$ Sec , $\tau = 0$ Sec , $E_t = 0$

Probability Threshold	Hint ที่ถูกต้อง	Hint ที่ผิด	Hint ที่ซ้ำซ้อน	จำนวน Request ที่สร้าง Hint	Recall	Precision	Average Hintsize
0.2	26062	158066	31506	48994	0.3723	0.1415	3.0805
0.3	22482	76349	16967	34224	0.3212	0.2275	1.6543
0.4	19605	47237	12144	29843	0.2801	0.2933	1.1284
0.5	18577	40154	10675	26990	0.2654	0.3163	0.9915
0.6	16509	24592	7593	24090	0.2358	0.4017	0.6956
0.7	14901	17325	6357	21083	0.2129	0.4624	0.5512
0.75	14286	14785	5140	18566	0.2041	0.4914	0.4887
0.8	12263	10859	3962	17414	0.1752	0.5304	0.3869
0.85	7114	7166	3231	9191	0.1016	0.4982	0.2502
0.9	5371	4935	2004	6330	0.0767	0.5212	0.1759
0.95	3185	3066	1212	2791	0.0455	0.5095	0.1066
1	2904	2728	1017	2401	0.0415	0.5156	0.0950

ตารางที่ 4.3 ผลการทดสอบที่ $T = 300$ Sec , $\tau = 0$ Sec , $E_t = 0$

Probability Threshold	Hint ที่ถูกต้อง	Hint ที่ผิด	Hint ที่ซ้ำซ้อน	จำนวน Request ที่สร้าง Hint	Recall	Precision	Average Hintsize
0.2	29222	255843	50175	52008	0.4175	0.1025	4.7891
0.3	23856	113599	24049	39205	0.3408	0.1736	2.3072
0.4	20752	63756	15409	32635	0.2965	0.2456	1.4274
0.5	19339	52787	12449	28869	0.2763	0.2681	1.2082
0.6	16928	29858	8546	25067	0.2418	0.3618	0.7905
0.7	15561	21209	6876	22517	0.2223	0.4232	0.6235
0.75	15008	19319	6424	21271	0.2144	0.4372	0.5822
0.8	13800	14017	4513	18165	0.1971	0.4961	0.4619
0.85	7743	9029	3644	9754	0.1106	0.4617	0.2917
0.9	7012	7150	2305	8331	0.1002	0.4951	0.2352
0.95	4088	4641	1624	3612	0.0584	0.4683	0.1479
1	3616	4129	1199	2701	0.0517	0.4669	0.1278

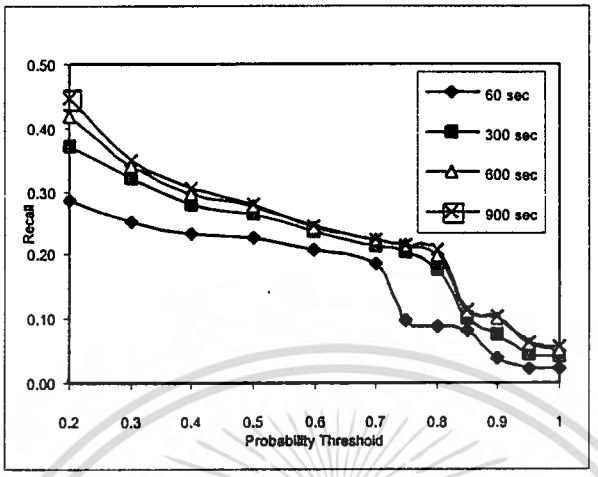
ตารางที่ 4.4 ผลการทดสอบที่ $T = 600$ Sec , $\tau = 0$ Sec , $E_t = 0$

Proability Threshold	Hint ที่ถูกต้อง	Hint ที่ผิด	Hint ที่ซ้ำซ้อน	จำนวน Request ที่สร้าง Hint	Recall	Precision	Average Hintsize
0.2	31232	289158	54497	54139	0.4462	0.0975	5.3555
0.3	24517	137323	27539	41131	0.3502	0.1515	2.7054
0.4	21364	79261	18470	36001	0.3052	0.2123	1.7014
0.5	19604	61051	13837	29339	0.2801	0.2431	1.3499
0.6	17114	34119	8809	26114	0.2445	0.3340	0.8577
0.7	15665	23852	6974	22956	0.2238	0.3964	0.6642
0.75	15042	21652	6639	21553	0.2149	0.4099	0.6190
0.8	14420	17657	4934	20622	0.2060	0.4495	0.5287
0.85	8008	10284	3771	9945	0.1144	0.4378	0.3152
0.9	7285	8239	2467	8501	0.1041	0.4693	0.2570
0.95	4456	5611	1784	3758	0.0637	0.4426	0.1693
1	3986	5105	1381	2923	0.0569	0.4385	0.1496

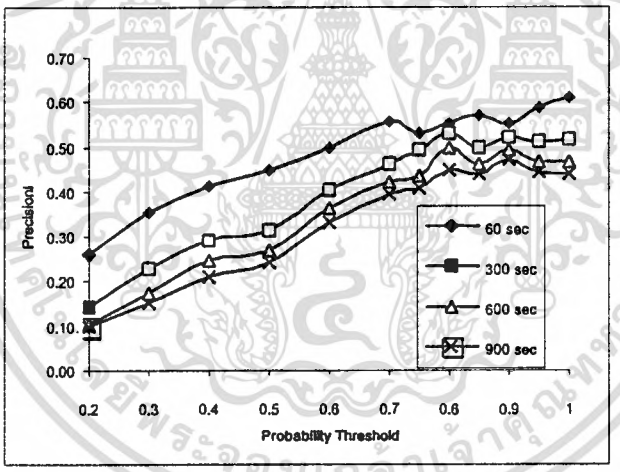
ตารางที่ 4.5 ผลการทดสอบที่ $T = 900$ Sec , $\tau = 0$ Sec , $E_t = 0$

Effective Threshold	Hint ที่ถูกต้อง	Hint ที่ผิด	Hint ที่ซ้ำซ้อน	จำนวน Request ที่สร้าง Hint	Recall	Precision	Average Hintsize
0.000	24134	109463	23586	39117	0.3448	0.1806	2.2455
0.025	21491	42151	5612	38863	0.3070	0.3377	0.9893
0.050	21382	41701	5093	38863	0.3055	0.3390	0.9739
0.075	20787	38084	4392	38814	0.2970	0.3531	0.9038
0.100	20638	37170	3326	38782	0.2948	0.3570	0.8733
0.150	20380	34723	2805	38578	0.2911	0.3699	0.8273
0.200	19769	32056	2152	38488	0.2824	0.3815	0.7711
0.250	19352	27938	1461	37426	0.2765	0.4092	0.6964
0.300	18696	23882	1139	34649	0.2671	0.4391	0.6245
0.400	18096	21317	801	33467	0.2585	0.4591	0.5745
0.500	17752	20278	623	32540	0.2536	0.4668	0.5522

ตารางที่ 4.6 ผลการทดสอบที่ $T = 300$ Sec , $\tau = 0$ Sec , $P_t = 0.25$

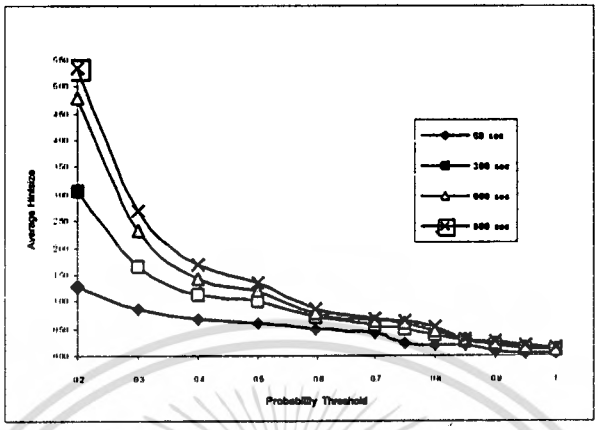


(a) Recall vs. Probability Threshold

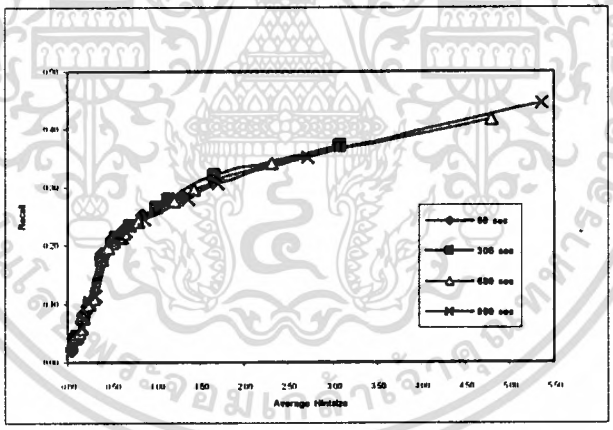


(b) Precision vs. Probability Threshold

รูปที่ 4.15 Recall และ Precision ด้วย Probability Threshold ที่ Maximum T ต่างๆ

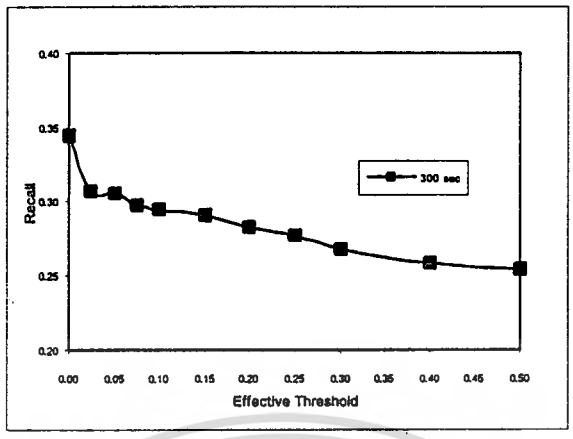


(a) Average hintsize vs. Probability Threshold

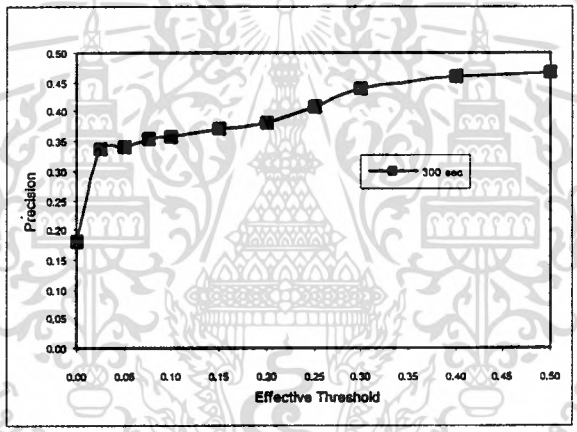


(b) Recall vs. Average hintsize

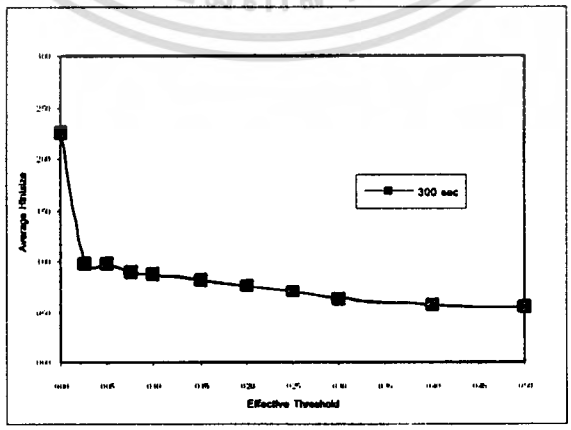
รูปที่ 4.16 Average hintsize ด้วย Probability Threshold และ Recall ที่ Maximum T ต่างๆ



(a) Recall vs. e_t



(b) Precison vs. e_t



(c) Average hintsize vs. e_t

รูปที่ 4.17 Recall , Precison และAverage hintsize ด้วย Effective Threshold

4.2.2 การวิเคราะห์ผลการทดสอบ

จากรูปที่ 4.15 – 4.16 แสดงถึงคุณสมบัติของ Pairwise Volume โดยจากรูปจะเห็นว่า Recall และ Hintsize ลดลง เมื่อเพิ่ม Probability Threshold สูงขึ้นซึ่งกล่าวได้ว่า Probability Threshold ที่สูงจะมีค่าใน Volume ที่ใช้ Probability Threshold ที่ต่ำกว่า ลักษณะการเพิ่มของ Precision ด้วยการลดลงของ Recall และ Hintsize เป็นคุณสมบัติของ Volume ที่ต้องการซึ่ง Pairwise Volume มีคุณสมบัตินี้ และ เมื่อพิจารณาที่เวลา T = 60 , 300 , 600 , 900 sec นั้นจะพบได้ว่าความสัมพันธ์ที่เกิดจาก Resource คู่หนึ่งๆ ส่วนใหญ่จะใช้เวลาประมาณ 600 sec โดยที่สังเกตได้ว่าผลที่ได้จากเวลา 900 sec ไม่แตกต่างจากเวลา 600 sec มากนัก

รูปที่ 14.15 (a) Recall จะลดลงในช่วงที่ Probability Threshold มีค่าสูงขึ้นแสดงว่าคำทำนายมีน้อยลงในช่วงดังกล่าวและจะมากขึ้นเมื่อ Probability Threshold น้อยลง ส่วนรูป 14.16 (a) แสดงให้เห็นว่าจำนวน Hint ของ Pairwise ที่มี Probability Threshold ต่ำจะมีเป็นจำนวนมาก รูป 14.16 (b) แสดงให้เห็นว่าช่วงเวลาที่พิจารณามากจำนวนความสัมพันธ์ของการสร้างคำทำนายก็ยิ่งมาก

จากการทดลองที่ให้ความสัมพันธ์กับ Resource ตัวท้ายจะได้ว่ารูปที่ 14.17 แสดงให้เห็นว่า e_t ที่ค่าสูงจะปรากฏอยู่ใน e_t ค่าต่ำกว่า และเมื่อเปรียบเทียบกับ กรณีที่กำหนด e_t เป็น 0 แล้วพิจารณาแต่ค่า Probability Threshold นั้นจะได้จำนวน Recall ที่ต่ำ แต่เมื่อนำ e_t มาพิจารณาจะพบว่าถึงแม้ใช้ Probability Threshold ค่าต่ำก็ยังให้ผลที่ Precision ที่สูง และ ค่า Recall ที่สูงกว่า

บทที่ 5
บทสรุป

5.1 สรุปผลการศึกษา

โปรแกรมการทำนายการเรียกใช้ข้อมูลที่พัฒนาขึ้นมานี้ เป็นโปรแกรมที่พยายามอาศัยข้อมูลเดิมที่ได้จัดเก็บอยู่แล้วไว้ในองค์กรมาใช้ให้เกิดประโยชน์โดยการนำประวัติการใช้งานที่ผ่านมามาพิจารณาหาความสัมพันธ์ของข้อมูลที่น่าจะเป็นไปได้

ในการศึกษานี้ได้ทำการเขียนโปรแกรมเพื่อจัดกลุ่มข้อมูลเพื่อพิจารณาถึงข้อมูลที่น่าจะถูกเรียกใช้ในลำดับถัด ซึ่งตัวโปรแกรมสามารถแบ่งเป็น 2 ขั้นตอนหลัก คือ การทำความสะอาดข้อมูล และ การหาความสัมพันธ์ในการเรียกใช้ข้อมูล โดยในส่วนแรกนั้นจะทำความสะอาดข้อมูลเนื่องจากข้อมูลดิบที่มีอยู่ซึ่งมีขนาดใหญ่มาก บางครั้งมีข้อมูลที่ซ้ำซ้อนหรือผิดพลาด และ มีข้อมูลบางส่วนไม่มีความสำคัญตามหลัก Zipf Law จึงพิจารณาข้อมูลที่เป็นส่วนใหญ่เท่านั้น เพราะข้อมูลที่เป็นส่วนย่อยนอกจากทำให้เสียเวลาในการคำนวณแล้วผลของส่วนย่อยนี้ก็ถูกนำมาพิจารณาน้อยมาก โดยผลที่ได้ส่วนนี้จะนำไปใช้ในขั้นตอนถัดไป จากที่ได้ศึกษามาได้เลือกใช้วิธี Pairwise Probabilty ซึ่งจะใช้เวลาในการคำนวณที่น้อยกว่า โดยที่ยังคงได้ผลลัพธ์ที่ดีตามเงื่อนไขของตัวแปรที่กำหนด ในที่นี้คือ การเลือกพิจารณาที่ ค่า Probability Threshold และ Effectiveness Threshold ต่างๆ โดยวัดผลลัพธ์ตามตัวแปร Recall , Precision และ Hint Size

จากการพัฒนาโปรแกรมตามอัลกอริทึมนี้สามารถนำหลักการนี้ไปใช้จัดกลุ่มข้อมูลใน Web Server เพื่อการทำนายจะสามารถนำไปใช้ในการ Prefetching เพื่อลด Preceived Latency ของผู้ใช้ หรือ ลดการใช้งานเครือข่ายโดยการจัด Cache Validataion Request และการจัดการ Cache Replacement ของ Proxy โดยอาศัยข้อมูลจาก Log ขนาดใหญ่พร้อมทั้งพิจารณาตามเกณฑ์ที่เหมาะสมตามที่กล่าวมาแล้ว

5.2 ข้อเสนอแนะ

โปรแกรมนี้ยังมีข้อจำกัดที่ควรจะต้องปรับปรุงแก้ไขเพื่อให้มีความยืดหยุ่นเหมาะสมกับองค์กรที่จะนำไปใช้ประโยชน์ ดังต่อไปนี้

- โปรแกรมนี้ใช้เวลาในการคำนวณ และ ทดสอบผลลัพธ์เป็นเวลานานมาก ยิ่งเมื่อต้องการข้อมูลในครบถ้วนก็ต้องยิ่งอาศัยข้อมูลจำนวนมาก

- ข้อมูลดิบที่นำมาใช้อาจจะไม่สมบูรณ์ครบถ้วนบ้าง เนื่องจากอาจจะเกิดมาจากลักษณะการบันทึกของตัว Web Server เอง หรืออาจจะมาจากการที่บ้างรายการการเรียกใช้ข้อมูลไม่ได้ทำกับ Server โดยตรงแต่ทำผ่านแคช Cache หรือ Proxy Server ทำให้พฤติกรรมของผู้ใช้ผิดพลาด หรือมาจากตัว เนื้อหาของ Web Page เองไม่ได้มีความสัมพันธ์กันภายใน Web Server นั้น

- ในการทดสอบเพื่อวัดผลนั้นอาจจะต้องใช้ข้อมูลหลากหลายคือ ต้องการทดสอบกับ Web Server ต่างชนิดกัน อาจทำให้ต้องเขียนโปรแกรมในบางขั้นตอนใหม่ เพื่อให้สอดคล้องกับรูปแบบการบันทึกของ Web Server นั้น ๆ

บรรณานุกรม

Bestavros, A.1995. "*Using speculation to reduce server load and service time on the WWW*" in Proc. ACM Inter. Conf. On Information and Knowledge Management.

Cohen, E. , Krishnamurthy, B. and Rexford, J. 1998. "*Efficient algorithms for prediction requests to web servers*" Tech. Rep981221-01, AT&T Labs – Research , December .

Cohen, E. , Krishnamurthy, B. and Rexford, J. 1998. "*Evaluating server-assisted cache replacement in the web*" in Proc. European Symposium on Algorithms, August .

Cohen, E. , Krishnamurthy, B. and Rexford, J. 1998. "*Improving end-to-end performance of the web using server volumes and proxy filters*" in Proc. ACM SIGCOMM , September .

Krishnamurthy, B. and Rexford, J. 1998 "*Software issues in characterizing web server logs*" in Proc. Web characterization Workshop, November.

Mogul, J. C. 1996. "*Hinted caching in the Web*" in Proc. ACM SIGOPS European Workshop.

Padmanabhan, V. N. and Mogul, J. C. 1996. "*Using predictive prefetching to improve World Wide Web latency*" Computer Communication Review, Vol. 26, no. 3, pp. 22-36.

ประวัติผู้เขียน

ชื่อผู้เขียน	นาย กิตติศักดิ์ ว่องไว
สถานที่เกิด	ภูเก็ต
วุฒิการศึกษาระดับปริญญาตรี	วิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมไฟฟ้า
สถานที่สำเร็จการศึกษา	มหาวิทยาลัยสงขลานครินทร์
ปีการศึกษาที่สำเร็จการศึกษา	ปีการศึกษา 2538