การแสดงผลลัพธ์การค้นหาข้อมูล: จุดเล็กๆ ที่สำคัญและไม่ควรมองข้ามสำหรับเสริชเอ็นจิน

The Search Results: The Importance Point that should not be Overlooked for Search Engines

ไกรศักดิ์ เกษร (Kraisak Kesorn)*

บทคัดย่อ

นักพัฒนาระบบเสริชเอ็นจิน (Search engine) ส่วนใหญ่มุ่งเน้นกลไกการค้นหาข้อมูล เช่น การทำดัชนี (Indexing) การเปรียบเทียบ (Matching) และการกรองข้อมูล (Filtering) เพื่อให้การค้นหามีความถูกต้องสูง แต่หลายคนละเลยที่จะให้ความสำคัญกับการแสดงผลลัพธ์จากการค้นหา แต่ความเป็นจริงแล้วส่วนการแสดงผลถือว่าเป็นส่วนที่สำคัญมากส่วนหนึ่ง เนื่องจากเป็นส่วนติดต่อกับผู้ใช้โดยตรง ดังนั้นแม้ว่าระบบมีกลไกการค้นหาข้อมูลที่ดีเยี่ยมเพียงใด แต่การแสดงผลข้อมูลที่ได้จากการค้นหาไม่มีประสิทธิภาพ ทำให้ผู้ใช้ต้องเสียเวลาค้นหาข้อมูลที่ต้องการจากกลุ่มข้อมูลผลลัพธ์ขนาดใหญ่ ปริมาณเป็นแสนหรือล้านเอกสาร ลักษณะนี้อาจจะเป็นสาเหตุหนึ่งที่ทำให้ระบบค้นหาข้อมูลนั้นไม่ได้รับการยอมรับจากผู้ใช้ ดังนั้นบทความนี้จะพูดถึงลักษณะการแสดงเอกสารผลลัพธ์ที่ได้จากการค้นหาในระบบเสริชเอ็นจิน โดยยึดหลักการเรียงลำดับเอกสารผลลัพธ์ (Ranking) การแสดงตัวอย่างเนื้อหาเอกสาร (Snippet) การออกแบบหน้าจอ (User Interface Design) นอกจากนั้นยังได้นำเสนอสิ่งสำคัญที่นักพัฒนาระบบควรคำนึงถึงเพื่อพัฒนาเสริชเอ็นจินในอนาคต

คำสำคัญ: การเรียงลำดับข้อมูล การคำนวณความคล้ายคลึง การแสดงตัวอย่างเอกสาร การสรุปเนื้อหาเอกสาร

* คณะวิทยาศาสตร์ มหาวิทยาลัยนเรศวร

Abstract

Many search engine developers focus mainly on indexing, matching and filtering data in the document collection in order to achieve a high retrieval performance. Unfortunately, the search result is overlooked by developers. In fact, displaying results to users is as important as other steps of the search mechanism because it is a vital part that communicates directly with users. Although a search engine has an exellent indexing, matching, or filtering mechanism, poor results lead to users wasting their time trying to find a desired document from the large number of documents in the result list and overlook the use of the search engine. This paper describes the various ideas in presenting the retrieved documents efficiently using the search engine. The main content includes ranking mechanism, text snippet, and user interface design. In addition, this paper presents the main issues that developers should consider to build an efficient search engine in the future.

Keywords: results ranking, similarity measurement, text snippet, personalized information retrieval.

1. บทนำ

ปัจจุบันเสริชเอ็นจินได้ทวีความสำคัญต่อชีวิตประจำวันของคนเราเกือบทุกคนในปัจจุบัน ทั้งในเรื่องการทำงาน การศึกษา ความบันเทิง หรือเรื่องการเมือง การที่มีข้อมูลจำนวนมากบนเครือข่ายอินเทอร์เน็ต นอกจากประโยชน์ทางด้านบวกคือการที่ผู้ใช้ค้นหาข้อมูลได้ง่ายภายในระยะเวลาอันสั้นแล้ว ยังมีผลเสียหรือด้านลบก็คือการที่เสริชเอ็นจินค้นหาเอกสารที่เกี่ยวข้องได้มากเกินไป การค้นหาบางครั้งได้ผลลัพธ์เอกสารเป็นจำนวนนับหมื่น แสน หรือเป็นล้านเอกสาร ซึ่งทำให้ผู้ใช้มีความรู้สึกเหมือนอยู่ในทะเลของเอกสารจำนวนมหาศาลและไม่รู้จะเลือกอ่านเอกสารไหนดี และไม่สามารถกรองเอกสารที่ไม่ต้องการออกไปได้ การเปิดเอกสารอ่านทีละเอกสารก็คงเป็นเรื่องเสียเวลามากสำหรับผู้ใช้ จากที่มาของปัญหาดังกล่าวจะเห็นว่าการแสดงผลลัพธ์ที่ได้จากการค้นหามีส่วนช่วยโดยตรงเพื่อช่วยให้ผู้ใช้สามารถกลั่นกรองเอกสารที่ไม่เกี่ยวข้องทิ้งไปได้ง่ายขึ้นและสามารถพบเอกสารที่ต้องการได้รวดเร็วมากยิ่งขึ้น บทความ

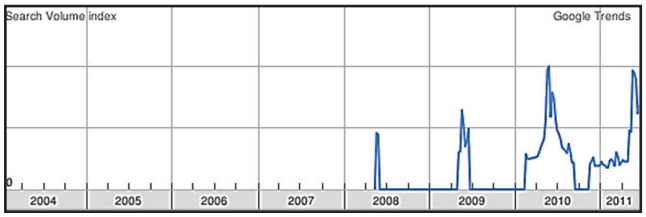
นี้ต้องการชี้ให้เห็นความสำคัญของกลไกการแสดงผลที่ได้จากการค้นหาข้อมูล ซึ่งเป็นสิ่งที่นักวิจัยหลายท่านมองข้ามไป นักวิจัยส่วนใหญ่มุ่งเน้นไปที่การทำดัชนี และการเปรียบเทียบความเหมือนระหว่างคิวรี่ (Query) และเอกสาร เป็นความจริงที่กลไกส่วนดังกล่าวมีความสำคัญต่อระบบเสริชเอ็นจินแต่หากว่าระบบไม่มีวิธีการแสดงผลการค้นหาที่ดีพอ ก็อาจจะทำให้ผู้ใช้ไม่เลือกใช้เสริชเอ็นจินดังกล่าวได้เช่นกัน

2. สิ่งสำคัญในการแสดงผลลัพธ์การค้นหาข้อมูลของเสริชเอ็นจิน

บทความวิชาการนี้จะสรุปหัวใจสำคัญของการแสดงผลลัพธ์การค้นหาข้อมูลของเสริชเอ็นจินตามประเด็นต่างๆ ดังต่อไปนี้ การเรียงลำดับเอกสารผลลัพธ์ (Result Ranking) การแสดงตัวอย่างเนื้อหาเอกสาร (Snippet) การออกแบบหน้าจอสำหรับเสริชเอ็นจินและ ความท้าทายในการออกแบบการแสดงผลการค้นหาของระบบเสริชเอ็นจินในอนาคต

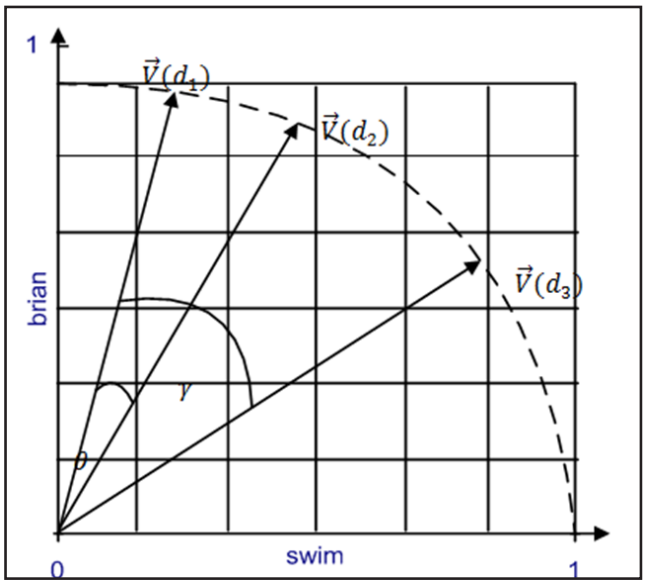
2.1 การเรียงลำดับเอกสารผลลัพธ์ (Result Ranking)

การค้นหาข้อมูลที่ได้จำนวนเอกสารปริมาณมากๆ การแสดงผลลัพธ์จำนวนมากให้กับผู้ใช้โดยไม่มีการคำนวณว่าเอกสารใดที่ผู้ใช้สนใจดูเหมือนจะไม่มีประสิทธิภาพนัก เนื่องจากการแสดงผลลัพธ์จำนวนมากๆ แก่ผู้ใช้ อาจทำให้ผู้ใช้สับสนและผู้ใช้ไม่สามารถกรองข้อมูลที่ไม่ต้องการออกไปได้ ดังนั้นจึงมีความจำเป็นที่ระบบการค้นหาข้อมูลจะต้องมีการเรียงลำดับผลลัพธ์ก่อนแสดงผลให้ผู้ใช้ทราบ จากการสำรวจพบว่านักวิจัยเริ่มให้ความสนใจในการพัฒนาระบบการจัดเรียงลำดับเอกสารเมื่อไม่กี่ปีที่ผ่านมา ภาพที่ 1 แสดงให้เห็นว่าเริ่มมีคนสนใจเกี่ยวกับการเรียงลำดับผลลัพธ์เอกสารเมื่อประมาณ 4 ปีที่แล้วมานี่เอง แต่แนวโน้มก็ชี้ให้เห็นว่าหัวข้อนี้ได้รับความสนใจจากนักวิจัยหรือนักพัฒนาระบบและถูกค้นหาข้อมูลมากขึ้นเรื่อยๆ ตามลำดับ



ภาพที่ 1 *แนวโน้มการค้นหาข้อมูลการเรียงลำดับผลลัพธ์*

เมื่อเสริชเอ็นจินทำการค้นหาข้อมูลได้แล้ว ระบบจะต้องตัดสินใจว่าจะนำเอกสารใดขึ้นเป็นลำดับแรก สอง และสามไปเรื่อยๆ ตามลำดับ หลักการจัดเรียงลำดับนี้ได้มาจากการคำนวณความคล้ายคลึง (Similarity) ระหว่างเอกสารและคิวรี่ ความคล้ายคลึงหมายถึงความเหมือนกันระหว่างเอกสาร 2 เอกสารโดยคำนวณออกมาเป็นตัวเลข ตัวเลขที่มีค่ามากแสดงว่าเอกสาร 2 เอกสารนั้นมีความคล้ายคลึงกันมากกว่าตัวเลขที่มีค่าน้อย วิธีการทั่วไปคือกลุ่มของเอกสารทั้งหมดในฐานข้อมูลจะถูกมองเป็นกลุ่มของเว็กเตอร์ในระบบเชิงพื้นที่เว็กเตอร์ (Vector space model) [1] เอกสารแต่ละเอกสารสามารถนำเสนอในรูปแบบของเส้นตรงที่ลากจากจุดโคออดิเนต (0,0) ไปยังจุดของเอกสารนั้นๆ บนพื้นที่เว็กเตอร์ ในการพิจารณาความเหมือนระหว่างเอกสาร 2 เอกสาร สามารถทำได้โดยหาความต่างระหว่างมุม (Magnitude different) ของเว็กเตอร์เอกสารบนพื้นที่เว็กเตอร์ ให้ *d1*, *d2*, และ *d3* หมายถึงเอกสารในฐานข้อมูล และ $\vec{v}(d_i)$ หมายถึงเว็กเตอร์ของเอกสารที่ i จากภาพที่ 2 สามารถสรุปได้ว่า $\vec{v}(d_1)$ เหมือน $\vec{v}(d_2)$ มากกว่า $\vec{v}(d_3)$ เนื่องจากค่าความต่างระหว่าง $\vec{v}(d_1)$ และ $\vec{v}(d_2)$ มีน้อยกว่า $\vec{v}(d_1)$ และ $\vec{v}(d_3)$ $(\theta < y)$ ในหัวข้อนี้จะแนะนำวิธีการคำนวณเพื่อหาค่าความคล้ายคลึงกันระหว่างคิวรี่และเอกสารโดยใช้หลักการของโมเดลเชิงพื้นที่เว็กเตอร์ โดยกำหนดให้ $Sin(x, y)$ คือค่าความคล้ายคลึงระหว่างเอกสาร x และ y โดยที่ $x = (x_1, x_2, x_3, ..., x_n)$ และ $y = (y_1, y_2, y_3, ..., y_n)$ การวัดความคล้ายคลึงจะพิจารณาจากค่าต่างๆ ที่ปรากฏในเอกสารในที่นี้ได้แก่ x_1-x_n และ y_1-y_n วิธีการที่นิยมได้แก่ Inner Product [1], Dice co-efficient [2],



ภาพที่ 2 *การเปรียบเทียบความเหมือนกันระหว่างเอกสารต่างๆ*

Cosine [3] และวิธี Distance [4] ในบรรดาเทคนิคเหล่านี้วิธีที่เป็นที่นิยมมากที่สุดคือวิธี Cosine และ Distance

Cosine similarity [3] เป็นวิธีการหาคล้ายคลึงกันจากค่าความต่างของมุมของวัตถุ 2 อันที่เกิดขึ้นบนพื้นที่เว็กเตอร์ (ดังแสดงในภาพที่ 2) ซึ่งความคล้ายคลึงกันแบบ Cosine นี้จะมีค่าอยู่ระหว่าง 0-1 เท่านั้น วิธีการนี้เป็นที่นิยมและมีประสิทธิภาพสูงในการวัดความคล้ายคลึงระหว่างวัตถุ 2 อัน และถูกนำมาประยุกต์ใช้กับศาสตร์การค้นคืนข้อมูลอย่างแพร่หลาย เนื่องจากวิธีการนี้จะมีประสิทธิภาพในกรณีที่เอกสาร 2 เอกสารมีความยาวไม่เท่ากันหรือทำให้มีความยุติธรรมต่อเอกสารที่สั้นกว่านั้นเอง

ส่วนการวัดความคล้ายคลึงโดยใช้ Distance [4] หรือระยะห่างระหว่างวัตถุ หลักการทำงานอยู่บนพื้นฐานที่ใช้ระยะทางระหว่างวัตถุ 2 อันที่อยู่ในพื้นที่เดียวกันเป็นตัวบอกความคล้ายคลึงกันของวัตถุ 2 อันนั้น ถ้าวัตถุอยู่ไกลกันมาก แสดงว่าวัตถุนั้นๆ มีความน่าจะเป็นที่จะมีคล้ายคลึงกันสูงกว่าวัตถุอื่นๆ ที่อยู่ไกลออกไปในพื้นที่เดียวกัน ดังนั้นเอกสารแต่ละเอกสารบนพื้นที่เว็กเตอร์สามารถวัดความคล้ายคลึงกันได้โดยดูจากระยะห่างระหว่างจุดบนพื้นที่เว็กเตอร์ การวัดระยะระหว่างเอกสาร 2 เอกสารคำนวณได้จากสมการที่ (1)

$$\delta(x,y)=\left(\sum_{i=1}^{n}(x_i-y_i)^k\right)^{\frac{1}{k}},\ k=1,2,...,\infty \qquad (1)$$

ทั้งนี้การคำนวณขึ้นอยู่กับค่าของ k เมื่อ k มีค่าเท่ากับ 1 จะเรียกสมการนี้ว่า การวัดระยะทางแบบแมนฮัทตัน (Manhattan distance measure) หรือ การวัดระยะทางแบบซิตี้บล๊อก (City block measure) หรือ การวัดระยะทางแบบแฮมมิ่ง (Hamming distance measure) เมื่อค่า k เท่ากับ 2 สมการที่ (1) จะกลายเป็นการวัดระยะทางแบบ Euclidean [5] ถ้าเอกสารใดๆ มีระยะทางไกลมากๆ (∞) หมายถึงความคล้ายคลึงกันของเอกสารจะมีค่าเข้าใกล้ 0 และถ้าระยะทางระหว่างเอกสารใกล้กันมากๆ ดังนั้นความคล้ายคลึงกันระหว่างเอกสารจะมีค่าเข้าใกล้ 1 อย่างไรก็ตามการคำนวณความคล้ายคลึงโดยพิจารณาเฉพาะความถี่ (Term frequency) ของคำในคิวรี่ที่ปรากฏในเอกสารต่างๆ อาจมีประสิทธิภาพไม่เพียงพอ เนื่องจากความยาวของเอกสารอาจจะทำให้มีความถี่ของคำต่างๆ มากกว่าเอกสารที่สั้นกว่า ดังนั้นอาจจะต้องมีการให้น้ำหนักของคำต่างด้วย เช่น วิธีการ *tf-idf* [1] วิธีการดังกล่าวถือว่ามีประสิทธิภาพสูงในการเรียงลำดับเอกสารผลลัพธ์ในปัจจุบัน แต่ในอนาคตแนวคิดเกี่ยวกับการค้นหาข้อมูลจะมีการนำความสนใจของผู้ใช้มาคำนวณร่วมด้วย ดังนั้นวิธีการดังกล่าวจึงต้องมีการปรับเปลี่ยนแนวคิดในการปรับเปลี่ยนนี้จะอธิบายเพิ่มเติมในหัวข้อ 2.4.1 และ 2.4.2

2.2 การแสดงตัวอย่างเนื้อหาเอกสาร

การเรียงลำดับเอกสารผลลัพธ์อาจจะไม่เพียงพอสำหรับผู้ใช้ในการกรองข้อมูล ดังนั้นนอกจากการแสดงผลลัพธ์ของการค้นหาตามความคล้ายคลึงของเอกสารต่อคิวรี่แล้ว สิ่งที่เสริชเอ็นจินควรทำอีกประการหนึ่งคือการแสดงข้อมูลที่เป็นประโยชน์ต่อผู้ใช้ เพื่อทำให้ผู้ใช้ทราบคร่าวๆ ว่าเอกสารนั้นๆ เกี่ยวข้องกับอะไร ปกติส่วนมากผู้ใช้จะไม่เปิดดูเอกสารทุกๆ เอกสารที่เป็นผลลัพธ์ของการค้นหา ดังนั้นระบบควรจะให้ข้อมูลคร่าวๆ สำหรับเอกสารแต่ละเอกสารเพื่อให้ผู้ใช้ตัดสินใจได้ว่า เอกสารนั้นน่าจะเกี่ยวข้องกับสิ่งที่ตนต้องการหรือไม่ วิธีมาตรฐานคือการแสดงตัวอย่างของเนื้อหา (Snippet) ในเอกสารให้ผู้ใช้ทราบ ซึ่งเป็นสรุปเนื้อหาอย่างย่อซึ่งทำให้ผู้ใช้ตัดสินใจได้ว่า เอกสารนั้นเกี่ยวข้องกับสิ่งที่ตนมองหาอยู่หรือไม่ โดยทั่วไปแล้วตัวอย่างของเนื้อหาในเอกสารประกอบด้วย ชื่อเอกสารและเนื้อหาสรุปอย่างย่อของเอกสารนั้น แต่คำถามที่ตามมาคือจะทำการสรุปเนื้อหาอย่างไรให้สามารถดึงใจความสำคัญของเอกสารออกมาและเป็นประโยชน์ต่อผู้ใช้ โดยทั่วไปการสรุปเนื้อหาของเอกสารมี 2 วิธีคือ วิธีที่ 1 การสรุปเนื้อหาแบบคงที่ (Static) ซึ่งไม่มีการเปลี่ยนแปลงใดๆ ตามคิวรี่ของผู้ใช้ และวิธีที่ 2 การสรุปเนื้อหาแบบไดนามิก (Dynamic) ซึ่งมีการเปลี่ยนแปลงตามเนื้อหาของคิวรี่ของผู้ใช้แต่ละคนและใช้อธิบายว่าทำไมเอกสารจึงถูกเลือกสำหรับคิวรี่นั้นๆ

2.2.1 การสรุปเนื้อหาแบบคงที่ (Static)

การสรุปเนื้อหาแบบคงที่ (Static) วิธีการที่ง่ายที่สุดสำหรับการสรุปเนื้อหาเอกสารแบบคงที่คือการแสดงตัวอย่าง 2 ประโยคแรกของเอกสาร หรือ 50 ตัวอักษรแรกของเอกสาร หรืออาจจะแสดงเนื้อหาส่วนใดส่วนหนึ่งของเอกสาร เช่น ชื่อเรื่องและชื่อผู้แต่ง ส่วนการใช้เมต้าเดต้า (metadata) ของเอกสาร เช่นการแสดงผู้แต่ง วันที่ เวลา เป็นต้น ในเอกสาร HTML ระบบอาจจะดึงข้อมูลที่อยู่ในส่วนของแท็กเมต้า (meta tag) มาแสดงเนื่องจากจะเป็นแหล่งเก็บข้อมูล

คำอธิบายสรุปของเนื้อหาในเว็บเพจนั้นๆ ข้อมูลในส่วนนี้จริงๆ แล้วถูกทำการดัชนีไว้แล้วทำให้การแสดงข้อความสรุปทำได้อย่างรวดเร็ว แทนที่จะไปดึงข้อความมาจากข้อความจากเอกสารจริง แต่ข้อเสียของวิธีการนี้คือ สิ่งที่สรุปอาจจะไม่ตรงกับสิ่งที่ผู้ใช้ต้องการ ดังนั้นวิธีการที่ดีกว่าคือการนำตัวประมวลผลภาษาธรรมชาติ (Natural Language Processing) มาช่วย ซึ่งสามารถจะสรุปเนื้อหาได้มีประสิทธิภาพมากกว่าวิธีการแรก อย่างไรก็ตามการทำงานยังอยู่บนพื้นฐานที่จะดึงเอาประโยคในเอกสารมาแสดงแต่จะเลือกประโยคที่สรุปเนื้อหาของเอกสารได้ดีที่สุด (อาจจะมากกว่า 1 ประโยค) ในวิธีการที่ซับซ้อนขึ้นไป อาจจะมีการตัดต่อประโยคใหม่ หรือสร้างเนื้อหาสรุปขึ้นมาใหม่สำหรับเอกสารนั้นๆ ซึ่งวิธีการนี้ได้รับความสนใจจากนักวิจัยอย่างมากในเรื่องเกี่ยวกับการสรุปข้อมูล (Text summarization) [6]

2.2.2 การสรุปเนื้อหาแบบไดนามิก (Dynamic)

การสรุปเนื้อหาแบบไดนามิก (Dynamic) มีการแสดงเนื้อหาเอกสารในส่วนที่ปรากฏคีย์เวิร์ดในคิวรี่เรียกว่า keyword-in-context การสรุปหาเนื้อแบบไดนามิกนี้จะทำงานร่วมกับการให้คะแนนเอกสารต่อคิวรี่ ถ้าคิวรี่เป็นวลี ความถี่ของวลีนั้นที่ปรากฏในเอกสารจะแสดงในส่วนของสรุป แต่ถ้าไม่เป็นแบบวลี ส่วนของเนื้อหาในเอกสารที่มีคำในคิวรี่ปรากฏอยู่เยอะๆ จะถูกเลือกมาแสดง การสรุปเนื้อหาแบบไดนามิกนี้เป็นที่นิยมกันในระบบค้นคืนข้อมูล แต่ข้อเสียคือมีความซับซ้อนในการทำงาน และไม่สามารถถูกคำนวณไว้ล่วงหน้าเนื่องจากไม่สามารถทราบได้ว่าคิวรี่ของผู้ใช้จะเป็นอะไร ระบบใดที่ใช้วิธีนี้อาจจะมีใช้เวลาในการตอบกลับ (Reponse time) ผู้ใช้นาน นอกจากนั้นความท้าทายของการสรุปเนื้อหาแบบไดนามิกคือ ต้องแสดงส่วนเนื้อหาที่เป็นประโยชน์ต่อผู้ใช้มากที่สุด อ่านเข้าใจง่าย และมีความยาวพอดี ไม่ยาวเกินพื้นที่สำหรับแสดงข้อมูลสรุปของเนื้อหาในหน้าจอผลลัพธ์ การสร้างเนื้อหาสรุปทั้งสองแบบจะต้องมีความเร็วในการทำงานที่ดีพอ เพื่อไม่ให้ไปกระทบต่อความเร็วในการแสดงผลการค้นหาต่อผู้ใช้ บางครั้งอาจจะมีการเก็บเอกสารไว้ในหน่วยความจำ (cache) แต่วิธีนี้ใช้ไม่ได้กับเอกสารที่มีความยาวมากๆ ซึ่งทำให้ไม่สามารถเก็บข้อมูลในหน่วยความจำที่จำกัดได้ ดังนั้นจึงต้องหลีกเลี่ยงไปใช้วิธีอื่นแทน เช่นเก็บข้อมูลเฉพาะบางส่วนในเอกสาร (Zone) ที่ใช้ในการทำสรุป เป็นต้น

ปัจจุบันการแสดงตัวอย่างของเอกสารผลลัพธ์ของ Google ได้พัฒนาขึ้นไปอีกขั้น ทั้งนี้เนื่องจากการแสดงตัวอย่างเนื้อหาแบบเดิมเป็นเพียงการตัดเนื้อหาบางส่วนที่มีคำในคิวรี่ปรากฏมาแสดงเท่านั้น ผู้ใช้อาจไม่แน่ใจว่าเอกสารนั้นเป็นเอกสารที่ตนเองต้องการหรือไม่ ดังนั้น Google ได้พัฒนาการแสดงตัวอย่างเอกสารไปอีกขั้น โดยอนุญาตเห็นตัวอย่างหน้าเว็บเพจของเอกสารนั้นได้ (ภาพที่ 3) วิธีการนี้ช่วยประหยัดเวลาและทำให้ผู้ใช้กรองข้อมูลได้ดีขึ้นโดยที่ผู้ใช้ไม่ต้องคลิกเพื่อเปิดเว็บเพจดังกล่าวๆ จริง

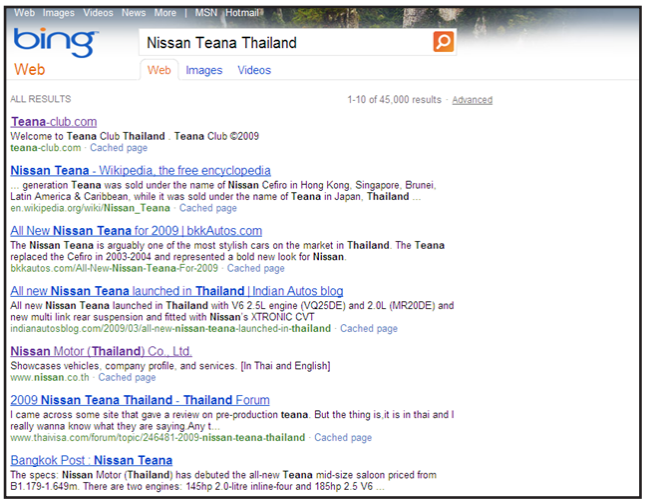
***ภาพที่ 3** ตัวอย่างแสดงเอกสารอย่างย่อของ Google*

2.3 การออกแบบหน้าจอสำหรับเสริชเอ็นจิน

นักวิจัยหลายท่านได้เสนอหลักการหรือแนวทางหลายๆ แนวทางในการออกแบบหน้าจอสำหรับซอฟท์แวร์ในระบบคอมพิวเตอร์ เช่น Shneiderman [7] ได้กำหนดแนวทางการออกแบบหน้าจอสำหรับระบบค้นหาข้อมูลสารสนเทศดังต่อไปนี้ 1) ระบบควรอนุญาตให้ผู้ใช้เห็นสิ่งที่กำลังมองหาได้ง่ายขึ้น 2) ระบบควรมีฟังก์ชันสำหรับผู้ใช้ที่มีความชำนาญสูง และ 3) ระบบควรลดข้อผิดพลาด หรือมีการจัดการข้อผิดพลาดได้

2.3.1 ระบบควรอนุญาตให้ผู้ใช้เห็นสิ่งที่กำลังมองหาได้ง่ายขึ้น

ตัวอย่างการช่วยให้ผู้ใช้เห็นสิ่งที่ตนเองกำลังมาหาได้ง่ายขึ้นได้แก่การไฮไลท์ (Highlight) หรือเน้นคำที่สัมพันธ์กับคำในคิวรี่ ผลลัพธ์ของการค้นหาข้อมูลในระบบปัจจุบันแสดงเป็นลำดับรายการจากบนลงล่าง ซึ่งแต่ละรายการจะประกอบด้วยข้อมูลแบบย่อ หัวเรื่อง (Title) หรือ URL เป็นต้น ซึ่งระบบควรจะทำการไฮไลท์คำดังกล่าวเพื่อให้ผู้ใช้หาข้อมูลที่ต้องการได้ง่ายขึ้น ในยุคแรกๆ ของเสริชเอ็นจินนั้น ระบบจะแสดงเนื้อหาแบบย่อของเอกสารที่เป็นผลลัพธ์ แต่ในปัจจุบันเสริชเอ็นจินจะทำการไฮไลท์เนื้อหาในส่วนที่มีคีย์เวิร์ดในคิวรี่ปรากฏอยู่ หรือเป็นคำที่มีความหมายใกล้เคียงกับคีย์เวิร์ดในคิวรี่

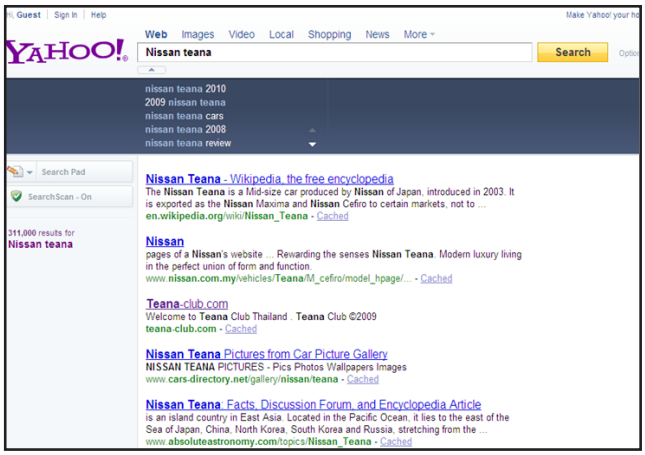


ภาพที่ 4 *ตัวอย่างหน้าจอแสดงผลลัพธ์การค้นหาข้อมูลของเว็บไซต์ Bing.com ซึ่งจะทำการไฮไลท์คีย์เวิร์ดในคิวรี่ที่ผู้ใช้ทำการค้นหา โดยคีย์เวิร์ดจะแสดงเป็นตัวหนาสีดำในส่วนของเนื้อหาที่มีความเกี่ยวข้องกับคีย์เวิร์ด*

เนื่องจากผู้ใช้มีความหลากหลายทั้งทางด้านความรู้ และอายุ ดังนั้นผู้ใช้ที่แตกต่างกันจึงมีการใช้คำศัพท์ที่หลากหลาย (Terminology heterogeneity) เพื่ออ้างถึงสิ่งเดียวกัน ดังนั้นระบบค้นหาข้อมูลควรจะมีการแนะนำคำศัพท์ที่ถูกต้องแก่ผู้ใช้ เช่น Google ซึ่งจะทำให้ระบบค้นหาข้อมูลได้ถูกต้องมากยิ่งขึ้น การแนะนำคีย์เวิร์ดที่นี้รวมถึงการแก้ไขคำที่สะกดผิด หรือการแนะนำคำศัพท์อื่นๆ (term expansion) ที่เกี่ยวข้องกับคีย์เวิร์ดที่ผู้ใช้ทำการค้นหาด้วยเช่นกัน หรือเป็นคำที่มีความหมายคล้ายคลึงกัน (Semantically similar words) ตัวอย่างการแนะนำคีย์เวิร์ดที่เกี่ยวข้องแสดงดังภาพที่ 5 ของเว็บไซต์ Yahoo เมื่อผู้ใช้พิมพ์ “Nissan Teana” Yahoo จะแนะนำคำศัพท์อื่นๆ ได้แก่ nissan teana 2010, 2009 nissan teana หรือ nissan teana cars แสดงในพื้นที่ด้านบนก่อนรายการผลลัพธ์การค้นหา เป็นต้น ซึ่งคำต่างๆ ที่จะถูกแนะนำนี้ขึ้นอยู่กับค่าทางสถิติที่ระบบเก็บไว้ เช่นค่า HITS

2.3.2 ระบบควรมีฟังก์ชันสำหรับผู้ใช้ที่มีความชำนาญสูง

ระบบควรมีฟังก์ชันเพิ่มเติมสำหรับผู้ที่มีความชำนาญในการค้นหาข้อมูลเสริชเอ็นจินส่วนใหญ่จะมี “Advanced search” เพื่อให้ผู้ใช้ระบุข้อมูลที่ต้องการค้นหาได้มากขึ้น ทำให้ระบบค้นหาข้อมูลได้ตรงกับสิ่งที่ผู้ใช้ต้องการ Advanced search คือการอนุญาตให้ผู้ใช้ใส่รายละเอียดในการค้นหาได้มากขึ้น เช่น ระบุภาษาของเอกสารที่ต้องการ ระบุเว็บไซต์ที่มีคำที่ไม่ต้องการ ประเภทของไฟล์ที่ต้องการค้นหา เป็นต้น อีกตัวอย่างของการมีฟังก์ชันสำหรับผู้ใช้ที่มีความชำนาญได้แก่ ระบบค้นหาข้อมูลรูปภาพที่อนุญาตให้ผู้ใช้สามารถที่จะวาดรูปภาพตัวอย่างลงบนหน้าจอ หรือเรียกว่า Query-by-Example และระบบจะทำการค้นหารูปภาพที่มีลักษณะคล้ายคลึงกับรูปภาพในคิวรี่ เป็นต้น แต่ข้อเสียประการหนึ่งของการที่ผู้ใช้สามารถระบุข้อมูลที่ต้องการค้นหาได้มากขึ้น อาจจะทำให้ระบบไม่สามารถค้นหาข้อมูลที่ตรงกับสิ่งที่ผู้ใช้ระบุได้ เนื่องจากคิวรี่มีความเฉพาะเจาะจงมากเกินไป



ภาพที่ 5 *ตัวอย่างการแนะนำคีย์เวิร์ดอื่นๆ ที่มีความเกี่ยวข้องกับคำศัพท์ในคิวรี่ของ Yahoo*

2.3.3 ระบบควรลดข้อผิดพลาด หรือมีการจัดการข้อผิดพลาดได้

การแก้ไขคำที่สะกดผิดของ Google เป็นตัวอย่างหนึ่งของการลดข้อผิดพลาด เพื่อหลีกเลี่ยงการค้นหาที่ใช้คีย์เวิร์ดที่สะกดผิดและทำให้เสริชเอ็นจินหาข้อมูลที่ผู้ใช้ต้องการไม่พบ ดังนั้นการที่ระบบมีฟังก์ชันช่วยในการสะกดคำหรือการขยายคีย์เวิรดไปยังคำอื่นๆ ที่เกี่ยวข้องจะช่วยป้องกันข้อผิดพลาดนี้ได้

2.4 ความท้าทายในการออกแบบการแสดงผลการค้นหาของระบบเสริชเอ็นจินในอนาคต

สำหรับการพัฒนาเสริชเอ็นจินในอนาคตนั้น มีสิ่งที่ท้าทายสำหรับนักวิจัยหรือผู้พัฒนาเสริชเอ็นจินอยู่หลายประการ ในบทความนี้จะชี้ความท้าทายที่สำคัญ 2 ประการได้ดังต่อไปนี้

2.4.1 การเรียงลำดับเอกสารผลลัพธ์เฉพาะบุคคล (Personalized ranking)

ทิศทางหนึ่งที่สำคัญของการพัฒนาระบบค้นหาข้อมูลบนอินเทอร์เน็ตคือ การค้นหาข้อมูลเฉพาะบุคคล [8] หมายถึงระบบค้นหาข้อมูลให้ตรงกับความต้องการของผู้ใช้แต่ละคน (Personalized Information Retrieval) มากยิ่งขึ้น นั่นหมายความว่าผู้ใช้ที่ใช้คิวรี่เหมือนกันแต่มีความสนใจต่างกัน เช่นชอบกีฬาฟุตบอลเหมือนกัน แต่คนหนึ่งชอบทีม Manchester United อีกคนหนึ่งชอบทีม Liverpool ดังนั้นผลลัพธ์ที่ได้จากการค้นหาคำว่า “football” จากผู้ใช้ 2 คน จะออกมาไม่เหมือนกัน นั่นคือการเรียงลำดับผลลัพธ์การค้นหาของแต่ละคนขึ้นอยู่กับผู้ใช้คนนั้นชอบหรือสนใจทีมใดเป็นพิเศษ ดังนั้นวิธีการคำนวณความคล้ายคลึงจะมีการปรับปรุงอย่างไร Castells et al. [9] ได้นำเสนอแนวคิดในการคำนวณวามคล้ายคลึงและเรียงลำดับผลลัพธ์โดยพิจารณาความสนใจของผู้ใช้ด้วย โดยโมเดลที่นำเสนอเรียกว่า “combSum” ซึ่งคำนวณจาก คำสำคัญในคิวรี่ (q) คำสำคัญในเอกสาร (d) และความสนใจของผู้ใช้ (u) และสามารถแสดงในรูปของสมการได้ดังนี้

$$score(d, q, u) = \lambda .prm(u, d)+(1-\lambda)sim(d, q) \qquad (2)$$

โดยที่ prm (u, p) คือ ความคล้ายคลึงระหว่างเอกสารและความสนใจของผู้ใช้ และ sim (d, q) คือ ค่าความคล้ายคลึงระหว่างเอกสารและ คิวรี่ รายละเอียดสามารถอ่านเพิ่มเติมได้ใน [9]

2.4.2 การเรียงลำดับเอกสารสำหรับระบบค้นคืนข้อมูลข้ามภาษา (Cross-Language Result Ranking)

แนวคิดที่น่าสนใจอีกประการหนึ่งคือการที่ทำให้เสริชเอ็นจินทำงานได้อย่างเต็มประสิทธิภาพและเป็นสากล (Universal) สำหรับผู้ใช้ที่ใช้ภาษาต่างกัน คือสามารถที่จะค้นหาเอกสารที่ถูกเขียนขึ้นภาษาหนึ่งโดยใช้คิวรี่ที่เขียนขึ้นคนละภาษากัน แต่ท้าทายที่เกิดขึ้นตามมาคือการคำนวณความเกี่ยวข้อง (Relevant) ระหว่างคิวรี่และเอกสารอื่นๆ ในคอลเล็คชั่นที่ถูกเขียนโดยใช้ภาษาที่ต่างกันจะมีหลักการคำนวณอย่างไร การเรียงลำดับเอกสารผลลัพธ์จะเรียงอย่างไร แนวคิดง่ายๆ ประการหนึ่งอาจจะเรียงลำดับโดยให้ความสำคัญกับเอกสารที่เป็นภาษาเดียวกับคิวรี่มากกว่าเอกสารอื่นๆ ที่เขียนคนละภาษากัน ตัวอย่างเช่น ถ้าผู้ใช้ค้นหาโดยใช้คิวรี่ภาษาไทยคำว่า “สุนัข” ระบบอาจจะเรียงลำดับเอกสารที่มีคำว่า “สุนัข” หรือ “หมา” ก่อนเอกสารที่มีคำว่า “dog” เป็นต้น

ตารางที่ 1 *เปรียบเทียบความสามารถของ เสริชเอ็นจิน 4 ค่ายหลักๆ ในปัจจุบัน*

Search engine	*Ranking*	*Snippet*	*Highlight Keywords*	*Advance Search*	*Keyword Suggestion*	*Personalized Ranking*	*Cross-Language Ranking*
Google	✓	✓	✓	✓	✓	×	×
bing	✓	✓	✓	✓	✓	×	×
YAHOO!	✓	✓	✓	✓	✓	×	×
Ask.com	✓	✓	✓	✓	✓	×	×

หมายเหตุ: ข้อมูล ณ วันที่ 28 มิถุนายน 2554

ความท้าทายเหล่านี้เป็นเพียงประเด็นหลักๆ ที่แสดงเป็นตัวอย่างในบทความนี้เท่านั้น ยังมีความท้าทายอื่นๆ อีกมากมาย เช่นการสรุปตัวอย่างเนื้อหาของเอกสารเพื่อให้ผู้ใช้ทราบว่าเนื้อหาของเอกสารอย่างคร่าวๆ โดยเฉพาะเอกสารภาษาไทย งานวิจัยทางด้านการสรุปเนื้อหาเอกสารภาษาไทยยังมีอยู่น้อยมาก จากความท้าทายต่างๆ ที่นำเสนอในบทความวิชาการนี้ สามารถสรุปเปรียบเทียบเสริชเอ็นจินได้ดังตารางที่ 1 จะเห็นได้ว่าเสริชเอ็นจินส่วนใหญ่รองรับการทำงานต่างๆ เกือบครบถ้วนอยู่แล้ว แต่อาจจะใช้อัลกอริทึมที่ต่างกัน เช่น Google ใช้ PigeonRank และBing ใช้ click-thru rate Yahoo ใช้ Yahoo Ranking Algorithm ในการเรียงลำดับเอกสาร แต่อย่างไรก็ตาม เสริชเอ็นจินเหล่านี้ยังไม่รองรับการทำงานการเรียงลำดับการเรียงลำดับเอกสารผลลัพธ์เฉพาะบุคคลและการเรียงลำดับเอกสารสำหรับระบบค้นคืนข้อมูลข้ามภาษา ดังนั้นสิ่งเหล่านี้จึงเป็นประเด็นที่น่าสนใจในการพัฒนาเสริชเอ็นจินในอนาคต

Zhou et al. [10] ได้เสนอวิธีการในการเรียงลำดับเอกสารของระบบค้นคืนข้อมูลข้ามภาษาอังกฤษและภาษาจีน โดยใช้หลักการของ Divergence from Randomness (DFR) โดยมีการคำนวณน้ำหนักของคำตามค่าการกระจาย (Distributed value) ในเอกสาร และในคอลเล็คชั่นของเอกสาร และจึงนำคำเหล่านั้นไปคำนวณความคล้ายคลึงระหว่างคิวรี่และเอกสารโดยใช้หลักการของ Latent Dirichlet Allocation (LDA) รายละเอียดสามารถอ่านเพิ่มเติมได้ใน [10] และ [11]

3. บทสรุป

กลไกการแสดงผลลัพธ์การค้นหาข้อมูลของเสริชเอ็นจินถือว่ามีความสำคัญไม่น้อยกว่ากลไกอื่นๆ ของระบบ เนื่องจากเสริชเอ็นจินที่ดีควรช่วยให้ผู้ใช้กรองข้อมูลจากผลลัพธ์ได้ง่าย ระบบต้องสามารถคำนวณความคล้ายคลึงระหว่างเอกสารและคิวรี่ได้อย่างมีประสิทธิภาพ และเรียงลำดับ (Rank) เอกสารผลลัพธ์จากความคล้ายคลึงมากไปหาน้อย นอกจากนี้เสริชเอ็นจินควรแสดงตัวอย่างเนื้อหาของเอกสารคร่าวๆ เพื่อช่วยให้ผู้ใช้เข้าใจว่าเอกสารแต่ละเอกสารแสดงเนื้อหาเกี่ยวกับอะไรโดยที่ผู้ใช้ไม่จำเป็นต้องเปิดอ่านเอกสารนั้นจริงๆ บทความนี้ชี้ทิศทางการพัฒนาการแสดงผลลัพธ์การค้นหาข้อมูลของเสริชเอ็นจินในอนาคตที่สำคัญ 2 ประเด็นคือ 1) การเรียงลำดับผลลัพธ์เฉพาะบุคคล และ 2) การคำนวณความคล้ายคลึงระหว่างเอกสารต่างๆ และคิวรี่ที่เขียนคนละภาษากัน ซึ่งเสริชเอ็นจินส่วนใหญ่ที่มีอยู่ในปัจจุบันยังไม่รองรับและรอการพัฒนาจากนักวิจัยต่อไป

4. เอกสารอ้างอิง

[1] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, London, United Kingdom: Cambridge University Press, 2008.

[2] G. Kondrak, D. Marcu, and K. Knight, "Cognates Can Improve Statistical Translation Models," *in Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pp. 46–48, 2003.

[3] P. N. Tan, M. Steinbach, and V. Kumar, "Introduction to Data Mining," 1st ed. Addison Wesley, 2005.

[4] E. F. Krause, "Taxicab Geometry: An Adventure in Non-Euclidean Geometry," Dover Publications, 1987.

[5] M. M. Deza and E. Deza, "Encyclopedia of Distances," *in Encyclopedia of Distances*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[6] V. A. Yatsko and T. N. Vishnyakov, "A Method for Evaluating Modern Systems of Automatic Text Summarization," *Automatic Documentation and Mathematical Linguistics*, vol. 41, no. 3, pp. 93-103, 2007.

[7] B. Shneiderman, "Designing the User Interface," 3rd ed. Addison Wesley, 1997.

[8] D. Vallet, P. Castells, M. Fernandez, P. Mylonas, and Y. Avrithis, "Personalized Content Retrieval in Context Using Ontological Knowledge," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 3, pp. 336-346, 2007.

[9] P. Castells, M. Fernández, D. Vallet, P. Mylonas, and Y. Avrithis, "Self-tuning Personalized Information Retrieval in an Ontology-Based Framework," *in OTM Workshops on the Move to Meaningful Internet Systems*, pp. 977-986, 2005.

[10] D. Zhou and V. Wade, "The Effectiveness of Results Re-Ranking and Query Expansion in Cross-Language Information Retrieval," *in Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering, and Cross-Lingual Information Access*, pp. 114-120, 2010.

[11] D. Zhou and V. Wade, "Latent Document Re-Ranking," *in Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, vol. 3, pp. 1571-1580, 2009.