บทที่ 2
วิธีการดำเนินการวิจัย

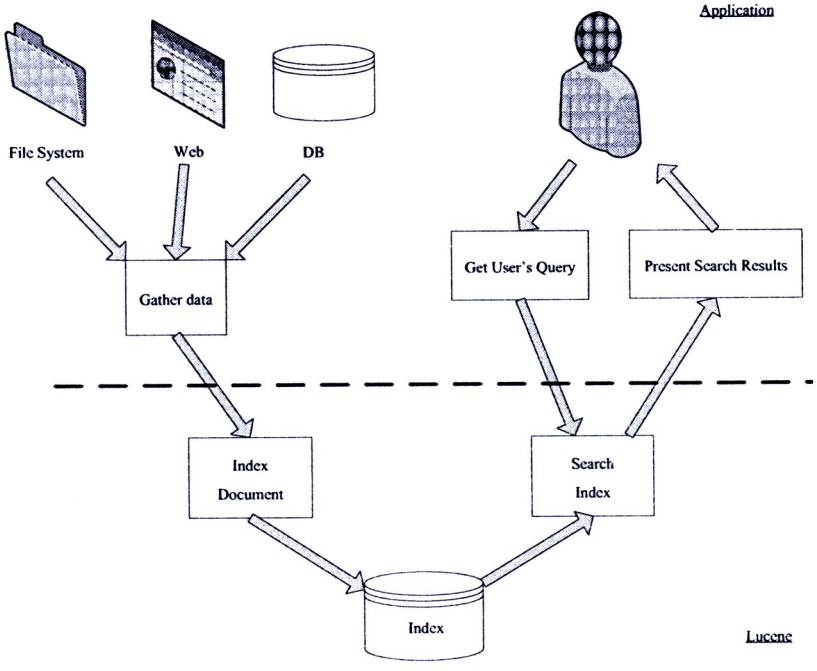
2.1 ลูซีน

ลูซีน (Lucene) เป็นโอเพนซอร์ส(Open Source Software) ที่มีเอพีไอ (API : Application Programming Interface) พร้อมให้นำไปประยุกต์ใช้ในแอพพลิเคชั่นสำหรับการค้นคืนข้อมูลและเอกสารในรูปแบบของตัวอักษร ซึ่งมีประสิทธิภาพในการสร้างดัชนี และค้นคืนได้อย่างรวดเร็ว และสามารถรองรับปริมาณเอกสารจำนวนมากได้ดีจึงทำให้ลูซีนได้รับการพัฒนาและนำไปใช้ในงานสำหรับการค้นคืนข้อมูลอย่างกว้างขวางและต่อเนื่อง

ผู้ที่ริเริ่มพัฒนาลูซีน คือ ดาว คัทติ้ง (Doug Cutting) โดยใช้ภาษาจาวาในการพัฒนาต่อมาในปี ค.ศ. 2001 ลูซีนได้รับความสนใจจากผู้ใช้มาก ทำให้ลูซีนถูกนำไปเป็นส่วนหนึ่งของโครงการจาการ์ต้า (Jakarta Project) ซึ่งอยู่ภายใต้โครงการซอฟต์แวร์อาพาเช่ (Apache Software Foundation) ในปัจจุบันได้มีการพัฒนาลูซีนในโปรแกรมภาษาต่างๆ มากมาย เช่น C, C++, C#, Perl และ Python

2.1.1 การประยุกต์ใช้ลูซีนสำหรับการพัฒนาระบบค้นคืนสารสนเทศ (Lucene for IR Application)

ลูซีนจะทำหน้าที่ในการสร้างฐานข้อมูลดัชนี และการค้นคืนเอกสารจากฐานข้อมูลดัชนีให้มีประสิทธิภาพมากที่สุด โดยคำนึงถึง เวลาที่ใช้ในการสร้างดัชนีและการค้นคืนเอกสาร (Indexing and Searching Time) ปริมาณเนื้อที่ที่ใช้ในการจัดเก็บดัชนี (Required Index Storage) ความถูกต้องและครอบคลุมในการค้นคืน (Precision and Recall)



รูปที่ 2.1 ภาพแสดงการนำลูซีนไปประยุกต์ใช้ในระบบค้นคืนสารสนเทศ

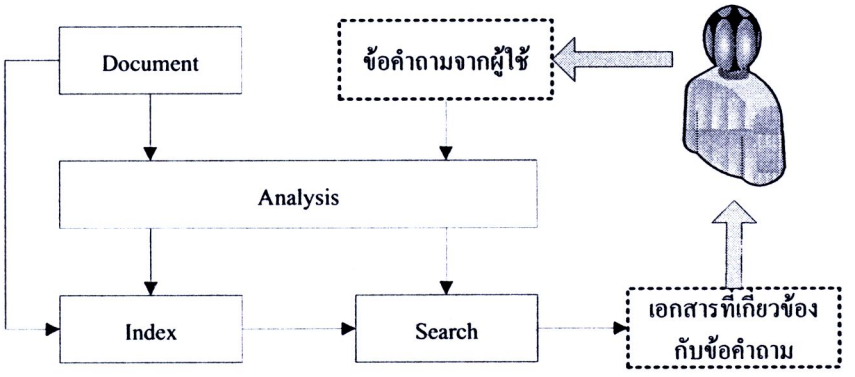
ในส่วนของนักพัฒนาโปรแกรมต้องทำหน้าที่ดึงและจัดเก็บเอกสาร ซึ่งข้อมูลนั้นอาจจะอยู่ในหลากหลายรูปแบบ และยังสามารถอยู่ในเครื่องคอมพิวเตอร์ที่ใช้งานอยู่หรืออาจอยู่ในเครื่องที่เชื่อมต่อในเครือข่ายก็ได้ เช่น ระบบไฟล์ ฐานข้อมูล หรือเอกสารบนเว็บ เป็นต้น อีกส่วนหนึ่งที่นักพัฒนาโปรแกรมต้องทำการพัฒนาขึ้นมาเองคือส่วนเชื่อมต่อระหว่างระบบกับผู้ใช้ซึ่งทำหน้าที่ในการรับข้อคำถามจากผู้ใช้ส่งผ่านไปให้ระบบประมวลผล และยังมีหน้าที่ในการแสดงผลลัพธ์จากการค้นคืนให้กับผู้ใช้ การออกแบบส่วนนี้ขึ้นอยู่กับลักษณะการใช้งานของระบบ แต่โดยทั่ว ๆไปปัจจัยหลักในการออกแบบคือ การคำนึงถึงผู้ใช้เป็นหลัก ระบบที่ดีควรจะให้ผู้ใช้สามารถค้นคืนได้ง่ายและสะดวกโดยไม่จำเป็นต้องเสียเวลาในการเรียนรู้การใช้งานมากนัก ซึ่งสามารถแสดงเป็นแผนภาพระหว่างส่วนที่ลูซีนรับผิดชอบและสิ่งที่ผู้พัฒนาระบบต้องทำเอง ดังรูปที่ 2.1

2.1.2 โครงสร้างทางสถาปัตยกรรมของลูซีน (lucene API)

ลูซีนมีโครงสร้างทางสถาปัตยกรรม 4 ส่วนที่สำคัญคือ

1. ส่วนการจัดการเอกสาร (Document)
2. ส่วนการวิเคราะห์คำ (Analysis)
3. ส่วนดัชนี (Index)
4. ส่วนการค้นคืน (Search)

สามารถที่จะนำมาเขียนเป็นรูปภาพแสดงการทำงานได้ดังรูปที่ 2.2



รูปที่ 2.2 การทำงานระหว่างสถาปัตยกรรมหลักของลูซีน

จากรูปที่ 2.2 กระบวนการทำงานระหว่างสถาปัตยกรรมจะเริ่มจากนำเอกสารที่เรามีมาเก็บไว้ใน ส่วนการจัดการเอกสารและกำหนดค่าฟิลด์ จากนั้นเอกสารที่จะนำมาสร้างเป็นดัชนีต้องนำไปผ่านในส่วนของการวิเคราะห์คำ เพื่อสกัดเอาคำที่สำคัญไปสร้างเป็นดัชนีซึ่งจะเก็บอยู่ในส่วนดัชนีรอผู้ใช้มาทำการค้นคืน หากผู้ใช้ต้องการทำการค้นคืนก็จะใส่ข้อคำถามมา ลูซีนจะทำการวิเคราะห์ข้อคำถามด้วยส่วนของการวิเคราะห์คำเช่นเดียวกับการวิเคราะห์เอกสารจากนั้นก็จะทำการค้นคืนเอกสารที่เกี่ยวข้องกับข้อคำถามด้วยส่วนการค้นคืนและแสดงเอกสารที่เกี่ยวข้องให้กับผู้ใช้โดยเรียงลำดับจากค่าความเกี่ยวข้องมากไปหาน้อย ซึ่งค่าความเกี่ยวข้องหาจากค่าคะแนนที่ได้

การทำงานของลูซีนในแต่ละส่วนมีการทำงานดังต่อไปนี้

1) ส่วนการจัดการเอกสาร

ส่วนการจัดการเอกสารมีหน้าที่ในการจัดโครงสร้างเอกสาร ข้อมูลที่จะนำมาเก็บในส่วนนี้จะต้องเป็นข้อความเท่านั้น ถ้าหากเป็นเอกสารที่อยู่ในรูปแบบเฉพาะ เช่น .pdf .doc จะต้องทำการแปลงรูปแบบก่อนจึงจะนำมาจัดเก็บได้ โดยผ่านทาง java.lang.String หรือ java.io.Reader ซึ่งแต่

ละเอกสารจะถูกแบ่งเป็นฟิลด์โดยจะประกอบไปด้วยฟิลด์เดียวหรือหลายฟิลด์ได้ไม่จำกัด จำนวน แล้วแต่รูปแบบของเอกสารนั้นๆ นอกจากนี้เราต้องทำการกำหนดรูปแบบของฟิลด์ เพื่อระบุให้การทำงานของคลาสต่อไปรู้ว่าจะต้องจัดการกับข้อมูลภายในฟิลด์อย่างไร ฟิลด์ประกอบไปด้วย 2 รูปแบบ คือ

1. อินเด็กซ์ (Index) เป็นการกำหนดค่าในฟิลด์นั้นให้นำไปทำเป็นดัชนีในการค้นคืน โดยสามารถกำหนดได้ด้วยว่าต้องการที่จะทำการวิเคราะห์ข้อความ หรือนอร์มัลไลซ์คำ (Normalize)
2. สโตร์ (store) เป็นการกำหนดค่าในฟิลด์นั้นว่าจะต้องเก็บข้อมูลเดิมหลังจากการทำดัชนีแล้วหรือไม่

ตารางที่ 2.1 ตารางระบุความหมายของฟิลด์อินเด็กซ์

Field.Index	ความหมาย
ANALYZED	ทำการวิเคราะห์คำก่อนทำเป็นดัชนี
ANALYZED NO NORMS	ทำการวิเคราะห์คำก่อนทำเป็นดัชนีแต่ไม่ต้องนอร์มัลไลซ์คำ
NO	ไม่ต้องทำเป็นดัชนี
NO NORMS	ทำเป็นดัชนีแต่ไม่ต้องนอร์มัลไลซ์คำ
NOT ANALYZED	ทำเป็นดัชนีแต่ไม่ต้องนอร์มัลไลซ์คำ
NOT ANALYZED NO NORMS	ทำเป็นดัชนีแต่ไม่ต้องวิเคราะห์และนอร์มัลไลซ์คำ

ตารางที่ 2.2 ตารางระบุความหมายของฟิลด์สโตร์

Field.Store	ความหมาย
COMPRESS	ทำการเก็บข้อมูลดั้งเดิมและบีบอัดข้อมูล
NO	ไม่เก็บข้อมูลดั้งเดิม
YES	ทำการเก็บข้อมูลดั้งเดิม

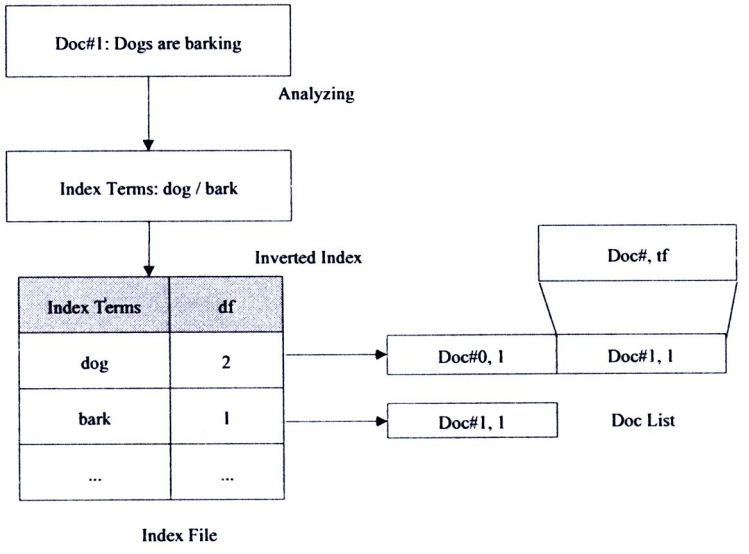
2) ส่วนการวิเคราะห์คำ

มีหน้าที่ในการวิเคราะห์ข้อความและสกัดคำสำคัญเพื่อนำไปสร้างเป็นดัชนี เช่น การลบอักขระพิเศษออก การเปลี่ยนตัวอักษรตัวใหญ่เป็นตัวเล็ก การแปลงคำให้อยู่ในรูปรากศัพท์ (Stemming) การตัดคำที่ไม่สำคัญออกได้ (Stopword Removal) เป็นต้น การเลือกรูปแบบของการวิเคราะห์คำเป็นสิ่งที่สำคัญอย่างมากในการพัฒนาระบบถ้าใช้การ วิเคราะห์คำไม่เหมาะสมก็จะทำให้ไม่สามารถค้นคืนเอกสารได้อย่างถูกต้องแม่นยำ การวิเคราะห์คำที่มีอยู่ในลูซีนมีอยู่ 4 แบบ คือ

1. การแบ่งเป็นคำตามช่องว่าง (WhiteSpaceAnalyzer) ซึ่งได้แก่ Space , Tab และ Newline Characters
2. การแบ่งเป็นคำตามช่องว่างและอักขระพิเศษที่ไม่ใช่ตัวอักษร (SimpleAnalyzer) เช่น Semi-Colon , Period , @ เป็นต้น รวมทั้งเปลี่ยนตัวอักษรภาษาอังกฤษตัวใหญ่เป็นตัวเล็ก
3. การแบ่งเป็นคำตามช่องว่างและอักขระพิเศษที่ไม่ใช่ตัวอักษร เปลี่ยนตัวอักษรภาษาอังกฤษตัวใหญ่เป็นตัวเล็กทั้งหมด รวมถึงการตัดคำที่มีอยู่ในเอกสารจำนวนมากแต่ไม่แสดงความหมายที่สำคัญ (Stopword) เช่น “a” , “an” , “the” , “but” , “it” เป็นต้น
4. การแบ่งเป็นคำตามช่องว่างและอักขระพิเศษที่ไม่ใช่ตัวอักษร เปลี่ยนตัวอักษรภาษาอังกฤษตัวใหญ่เป็นตัวเล็กทั้งหมด ตัดคำที่มีอยู่ในเอกสารมากแต่ไม่แสดงความหมายที่สำคัญ รวมถึงมีการวิเคราะห์หลักไวยากรณ์เพิ่มเติม คือ สามารถรู้ลักษณะที่อยู่ อีเมล์ ที่อยู่เว็บ ไอพีแอดเดรส ตัวย่อ และตัวอักษรที่ประกอบกับตัวเลข

3) ส่วนดัชนี

มีหน้าที่สร้างดัชนีจากคำที่ผ่านการวิเคราะห์คำมาแล้วหรือคำที่อยู่ในเอกสารที่ถูกกำหนดให้ไม่ต้องผ่านการวิเคราะห์คำ จากนั้นจึงนำมาเก็บแบบแฟ้มข้อมูลผกผันคือ แต่ละคำจะมีข้อมูลซึ่งระบุรายการหมายเลขของเอกสาร พร้อมทั้งจำนวนคำที่ปรากฏอยู่ในเอกสาร ดังรูปที่ 2.3



รูปที่ 2.3 การเก็บข้อมูลแบบแฟ้มข้อมูลผกผัน

จะเห็นได้ว่า คำว่า “dog” ซึ่งเป็นดัชนี มีจำนวนเอกสารที่มีคำว่า “dog” ทั้งหมด สองเอกสาร โดยเอกสารหมายเลข 0 และหมายเลข 1 มีคำว่า “dog” เอกสารละหนึ่งคำ

ค่า df หมายถึง จำนวนเอกสารที่มีคำนั้นอยู่

ค่า tf หมายถึง ความถี่ของคำที่อยู่ในเอกสารนั้น

โดยเราทำการสร้างดัชนีเพื่อเป็นตัวแทนเอกสารในการค้นคืนสารสนเทศ ทำให้การค้นคืนสามารถทำได้อย่างรวดเร็ว

4) ส่วนการค้นคืน

ทำหน้าที่ในการค้นคืนข้อมูลโดยหาเอกสารที่เกี่ยวข้องกับข้อคำถามของผู้ใช้จากการเปรียบเทียบกับดัชนีที่สร้างไว้ โดยการทำงานเริ่มจาก เมื่อได้รับข้อคำถามจากผู้ใช้ จะส่งต่อไปให้ คิวรี่พาร์เซอร์ (QueryParser) เพื่อทำการวิเคราะห์ข้อคำถาม จากนั้นทำการเปรียบเทียบข้อคำถามกับดัชนี และให้คะแนนเอกสารจากการคำนวณค่าคะแนนและจัดเรียงเอกสารที่เกี่ยวข้องจากค่ามากไปน้อยโดยใช้ ท็อปด็อก (TopDocs) ดังรูปที่ 2.4

รูปที่ 2.4 ขั้นตอนการค้นคืน

1. คิวรี่พาร์เซอร์ (QueryParser) ทำหน้าที่ในการประมวลผลและแปลงข้อคำถามก่อนการค้นคืน โดยจะใช้รูปแบบเดียวกับการวิเคราะห์เอกสารเพื่อให้ข้อความและดัชนีอยู่ในรูปแบบเดียวกัน
2. คิวรี่ (Query) ทำหน้าที่ค้นหาเอกสารที่เกี่ยวข้องกับข้อคำถามจากผู้ใช้ โดยมีหลายรูปแบบ
   - TermQuery ค้นหาโดยการเปรียบเทียบว่ามีคำอยู่ในเอกสารหรือไม่
   - RangQuery ค้นหาจากช่วงของดัชนี โดยระบุค่าเริ่มต้นและสิ้นสุด
   - PrefixQuery ค้นหาเอกสารทั้งหมดที่ขึ้นต้นด้วยคำที่ระบุในข้อคำถาม
   - BooleanQuery ค้นหาจากกลุ่มของคำที่มีเครื่องหมายตรรกะเป็นตัวเชื่อม ได้แก่ AND OR และ NOT
   - PhraseQuery ค้นหาจากกลุ่มของคำ
   - WildcardQuery ค้นหาโดยใช้ส่วนหนึ่งของคำหรือวลี
   - FuzzyQuery ค้นหาคำที่ใกล้เคียงกับข้อคำถาม โดยอาศัยการคำนวณระยะความแตกต่างของตัวอักษร
3. ท็อปด็อก (TopDocs) ใช้แสดงผลลัพธ์จากการค้นคืนสารสนเทศ โดยจะทำการเรียงลำดับเอกสารที่ทำการค้นคืนมาได้จากเอกสารที่มีค่าความเกี่ยวพันธ์มากไปหาน้อย โดยค่าความเกี่ยวพันธ์นี้มาจากค่าคะแนนที่ได้จากการคำนวนระหว่างเอกสารและข้อคำถาม ซึ่งมีการใช้ค่าต่างๆดังต่อไปนี้
   - Coord $_{q,d}$ คือ ค่าที่แสดงว่าเอกสารนี้มีจำนวนข้อคำถามในเอกสารมากหรือน้อย หากว่าเราค้นหาเอกสารด้วยข้อคำถามที่มี 2 คำเอกสารที่มีจำนวนข้อคำถามทั้ง 2 คำ จะมีค่าคะแนนมากกว่าเอกสารที่มีเพียงคำใดคำหนึ่ง และ หากเอกสารไม่มีข้อคำถามก็จะมีค่าคะแนนเป็นศูนย์

$$Coord = \frac{overlap}{\max Overlap} \qquad (2.1)$$

Overlap เป็นจำนวนข้อคำถามที่เอกสารมี
MaxOverlap เป็นจำนวนข้อคำถามทั้งหมด

- Sum of Squared Weights เป็นผลรวมที่คำนวณจากน้ำหนักของข้อคำถามแต่ละคำ

$$sumOfSquared = q.getBoost()^2 \times \sum_{t,d} idf_t \times t.getBoost()^2 \quad (2.2)$$

g.getBoost()^2 ค่าน้ำหนักของข้อคำถาม

t.getBoost()^2 ค่าน้ำหนักของคำแต่ละคำในข้อคำถาม

idf_t จำนวนเอกสารทั้งหมดที่มีข้อคำถาม t

- QueryNorm คือ ค่าที่ทำการนอร์มัลไลซ์ค่าน้ำหนักของข้อคำถาม หากข้อคำถามนั้นมีหลายคำ และแต่ละคำมีค่าน้ำหนักไม่เท่ากัน

$$queryNorm(q) = \frac{1}{\sqrt{sumOfSquaredWeight}} \quad (2.3)$$

- $tf_{t,d}$ ค่าความถี่ของข้อคำถาม t ที่ปรากฏในเอกสาร d

$$tf_{t,d} = \sqrt{frequency} \quad (2.4)$$

frequency เป็นความจำนวนข้อคำถาม t ที่อยู่ในเอกสาร d

- idf_t จำนวนเอกสารทั้งหมดที่มีข้อคำถาม t

$$idf = 1 + \log \frac{numDocs}{docFreq + 1} \quad (2.5)$$

numDocs เป็นจำนวนเอกสารทั้งหมด

docFreq เป็นจำนวนเอกสารทั้งหมดที่มีข้อคำถาม t

- length norm จำนวนคำในเอกสาร d

$$lengthNorm = \frac{1}{\sqrt{numterm}} \quad (2.6)$$

- $norm_{t,d}$ เป็นค่านอร์มัลไลซ์ระหว่างค่าน้ำหนักและความยาวเอกสาร

$$norm_{t,d} = doc.getBoost() \times lengthNorm(field) \times \prod_{field.f\ in\ d\ name\ as\ t} f.getBoost() \quad (2.7)$$

doc.getBoost() ค่าน้ำหนักของเอกสาร d

f.getBoost() ค่าน้ำหนักของ Field

length norm จำนวนคำในเอกสาร d

ซึ่งค่าสกอร์ของแต่ละเอกสารกับข้อคำถามจะมีค่าดังสมการที่ 2.8

$$score_{q,d} = coord_{q,d} \times queryNorm(q) \times \sum_{t,d} tf_{t,d} \times idf_t \times t.getBoost() \times norm(t,d) \quad (2.8)$$

2.2 การเลือกเทอมของเวกเตอร์โมเดล

2.2.1 การเลือกเทอม (Sort order)

หลักการนี้พัฒนาโดย ดอนน่า ฮาแมน (Donna Harman) มีแนวคิดที่ว่า จะหาข้อคำถามใหม่ที่ช่วยให้การค้นหาให้มีประสิทธิภาพมากขึ้นได้อย่างไร โดยการนำเอาหลักการของการค้นคืนย้อนกลับ (Relevance Feedback) รูปแบบของเทอม (Term variation) และการหาเส้นทางของเทอมที่อยู่ใกล้กันที่สุด (Nearest neighbor routines) มาทำการวิเคราะห์ โดยมีหลักในการคำนวณดังนี้

1) การหาค่านอยซ์เมทชัวร์ (Noise measure) n_k สำหรับแต่ละเทอมของเอกสารใดๆ หาได้จาก

$$n_k = \sum_{i=1} N \times \frac{tf_{ik}}{f_k} \times \log(tf_{ik} f_k) \tag{2.9}$$

โดยที่ N คือ จำนวนเทอมทั้งหมดของชุดเอกสาร

tf_{ik} คือ ความถี่ของเทอมที่ k ที่ปรากฏในเอกสารที่ i

f_k คือ ความถี่ของเทอมที่ k ที่ปรากฏในชุดเอกสาร

2) ฮาแมนมีการกำหนดสมมติฐานในการเลือกเทอมโดยมีสมมติฐานดังนี้

2.1) Noise: เป็นการใช้ค่านอยซ์เมทชัวร์มาทำการพิจารณาเลย

$$s(t_k) = \frac{1}{n_k} \tag{2.10}$$

2.2) Number of posting: เป็นการหาค่าสัมบูรณ์ของจำนวนเอกสารที่ค้นคืนได้ที่มีเทอมนั้นปรากฏอยู่

$$s(t_k) = \left|R^{t_k}\right| \tag{2.11}$$

2.3) Noise within posting: เป็นการหาค่านอยซ์เมทชัวร์ที่มาจากเอกสารที่ค้นคืนได้ที่มีเทอมนั้นปรากฏอยู่

$$s(t_k) = \frac{1}{n_k'} \tag{2.12}$$

2.4) Noise * tf_{ik} within posting: เป็นการหาค่านอยซ์เมทชัวร์ของเทอมนั้น คูณกับความถี่ของเทอมนั้นที่อยู่ภายในเอกสารที่ค้นคืนได้

$$s(t_k) = \frac{1}{n_k} df_k' \tag{2.13}$$

2.5) Noise * tf_{ik} * posting: เป็นการหาค่านอยซ์เมทชัวร์ของเทอมนั้น คูณกับความถี่ของเทอมนั้นที่ปรากฏในเอกสารทั้งหมด คูณกับจำนวนเอกสารที่ค้นคืนได้ที่มีเทอมนั้นปรากฏอยู่

$$s(t_k) = \frac{1}{n_k} df_k \left|R^{t_k}\right| \tag{2.14}$$

2.6) Noise *tf_{ik} : เป็นการหาค่านอยซ์เมทชัวร์ของเทอมนั้น คูณกับความถี่ของเทอมนั้นที่ปรากฏในเอกสารทั้งหมด

$$s(t_k) = \frac{1}{n_k} df_k \tag{2.15}$$

2.3 วีไอพีเอส (Vision Based Pages Segmentation : VIPS)

ปัจจุบันเว็บกลายเป็นแหล่งข้อมูลที่ใหญ่ที่สุดในการค้นหาข้อมูล ซึ่งในการทำการค้นหาจะค้นหาไปทีละหน้าเว็บของแต่ละเว็บนั้น โดยที่ในแต่ละหน้าเว็บนั้นส่วนมากจะประกอบด้วยหลายๆส่วนประกอบที่อาจจะไม่เกี่ยวข้องกับเนื้อหาที่เราสนใจ เช่น โฆษณา แถบเมนูต่างๆ ทำให้การค้นคืนได้ในสิ่งที่ไม่ตรงกับที่เราต้องการ ดังนั้นการนำอัลกอริทึมวีไอพีเอสมาใช้ในการแบ่งหน้าเว็บเพจให้เป็นบล็อก เพื่อจำกัดการค้นคืนเฉพาะส่วนที่เป็นเนื้อหาสำคัญ

วีไอพีเอสเป็นอัลกอริทึมที่ใช้ในการแบ่งโครงสร้างทางภาษาของเว็บเพจ ซึ่งในโครงสร้างทางภาษานั้นมีลักษณะเป็นลำดับชั้นและมีโหนดตามลำดับชั้น โดยแต่ละโหนดนั้นจะถูกเรียกว่าบล็อก และจะถูกกำหนดค่าดีโอซี (Degree of Coherence: DoC) เพื่อแสดงว่าแต่ละส่วนในบล็อกนั้นจะถูกรวมกันได้อย่างไร

2.3.1 โครงสร้างเนื้อหา (Vision Based Content Structure)

ในโครงสร้างเนื้อหานั้นจะมีการกำหนดให้ทุกโหนดของโครงสร้างนี้เรียกว่าบล็อก โดยแต่ละโหนดในโครงสร้างนี้ไม่จำเป็นต้องตรงกับทุกโหนดในดอมทรี ซึ่งสามารถอธิบายโมเดลของโครงสร้างของเนื้อหาได้จากส่วนประกอบของเว็บเพจสามารถแทนด้วยสมการต่อไปนี้

$$\Omega = (\mathrm{O}, \Phi, \delta) \tag{2.16}$$

โดยที่ Ω คือ ส่วนประกอบของเว็บเพจ

O คือ เซตจำกัดของทุกบล็อกแต่ทุกบล็อกจะไม่ซ้อนทับกันและ $\mathrm{O} = \{ \Omega^1, \Omega^2, \ldots, \Omega^N \}$

Φ คือ เซตจำกัดของตัวแบ่ง และ $\Phi = \{ \varphi^1, \varphi^2, \ldots, \varphi^T \}$

δ คือ ความสัมพันธ์ของทุกๆ 2 บล็อกใน O

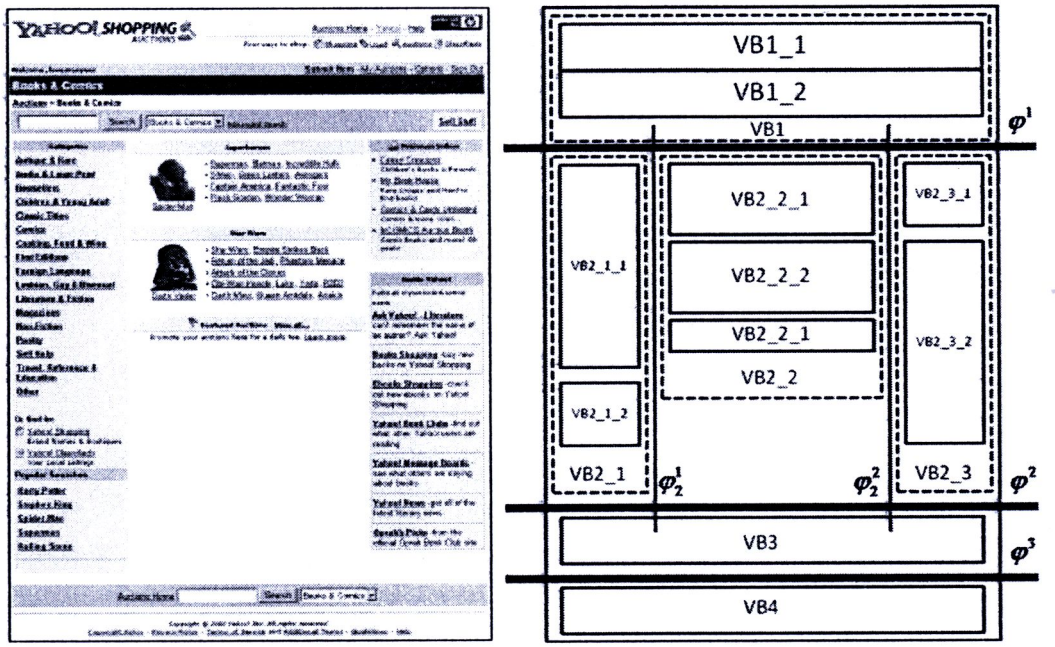
จากรูปที่ 2.5 แสดงตัวอย่างหน้าเว็บและโครงร่างเว็บเพจของ Yahoo! Shopping Auctions ที่ถูกแบ่งโดยโครงร่างเว็บเพจในระดับแรกหน้าเว็บที่เราได้มาสามารถแบ่งเป็น 4 บล็อกคือ VB1 - VB4 และได้ 3 ตัวแบ่งคือ $\varphi^1 - \varphi^3$ ดังรูป จากนั้นเราสามารถแบ่งโครงสร้างได้ย่อยต่อไปอีก เช่น VB2 นั้นสามารถแบ่งได้อีกเป็น 3 บล็อกและ 2 ตัวแบ่ง จะเห็นว่าโครงสร้างของเนื้อหา เป็นการแบ่งโดยใช้ในการแบ่งเนื้อหาที่แตกต่างกันออกจากกัน จากรูปที่ 2.5 นี้จะเห็นว่า VB2_1_1 เป็น category ลิงค์ของ Yahoo Shopping Auctions และ VB2_2_1 และ VB2_2_2 แสดงรายละเอียดของตัวการ์ตูน ซึ่งแต่ละส่วนสามารถแสดงตัวอย่างรายละเอียดของโครงสร้างเนื้อหาของเว็บ Yahoo! Shopping Auctions ได้ดังสมการต่อไปนี้

$$\mathrm{O} = (VB1, VB2, VB3, VB4) \tag{2.17}$$

$$\Phi = \{\varphi^1, \varphi^2, \varphi^3\} \tag{2.18}$$

$$\delta\begin{pmatrix}(VB1,VB2)\\(VB2,VB3)\\(VB3,VB4)\\else\end{pmatrix}=\begin{pmatrix}\varphi^1\\\varphi^2\\\varphi^3\\NULL\end{pmatrix} \tag{2.19}$$

จากสมการ 2.17 แสดงว่าเว็บเพจนี้แบ่งออกได้เป็น 4 บล็อกคือ VB1, VB2, VB3 และ VB4 และสมการ 2.18 จะเห็นว่ามี 3 ตัวแบ่ง คือ φ^1, φ^2, φ^3 และสมการ 2.19 ใน δ สามารถบอกได้ว่าแต่ละตัวแบ่งนั้นแบ่งระหว่างบล็อกใด เช่น φ^1 นั้นจะแบ่งระหว่างบล็อก VB1 กับ VB2 เป็นต้น



รูปที่ 2.5 ตัวอย่างหน้าเว็บของ Yahoo! Shopping Auctions และโครงร่างของหน้าเว็บเพจที่ถูกแบ่ง

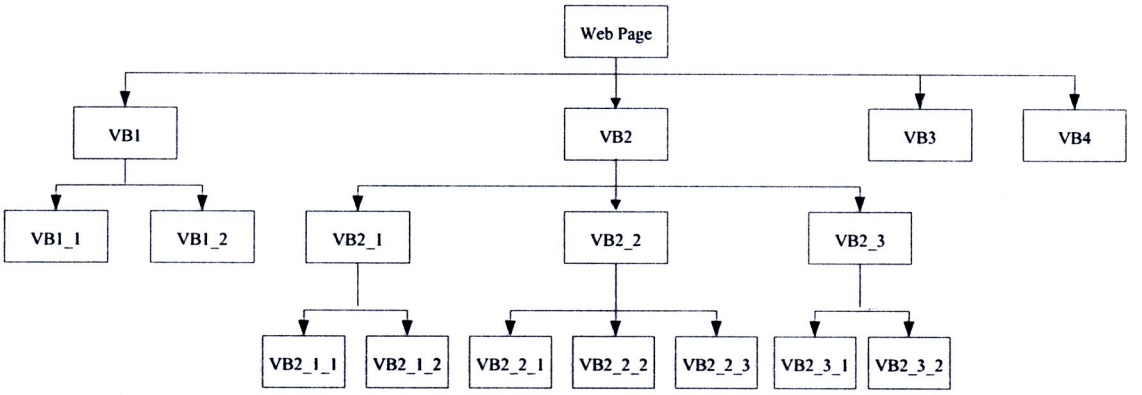
$$VB2=(VB2\_1,VB2\_2,VB2\_3) \tag{2.20}$$

$$\Phi^2=\{\varphi_2^1,\varphi_2^2\} \tag{2.21}$$

$$\delta^2=\begin{pmatrix}(VB2\_1,VB2\_2)\\(VB2\_2,VB2\_3)\\else\end{pmatrix}=\begin{pmatrix}\varphi_2^1\\\varphi_2^2\\NULL\end{pmatrix} \tag{2.22}$$

จากสมการ 2.20 แสดงรายละเอียดโครงสร้างเนื้อหาของบล็อก VB2 ว่าแบ่งย่อยอีกได้เป็น 3 บล็อก คือ VB2_1, VB2_2, VB2_3 จากสมการ 3.21 จะเห็นว่ามี 2 ตัวแบ่งคือ φ_2^1 และ φ_2^2 และสุดท้ายจากสมการ

2.22 แสดงว่าตัวแบ่ง φ_2^1 นั้นแบ่งระหว่างบล็อก VB2_1 และ VB2_2 และตัวแบ่ง φ_2^2 แบ่งระหว่างบล็อก VB2_2 และ VB2_3

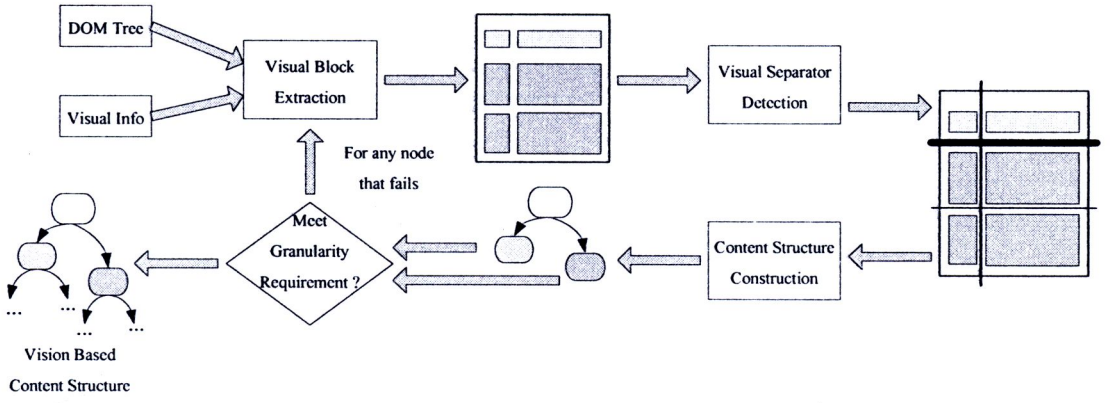


รูปที่ 2.6 โครงสร้างเนื้อหาของเว็บ Yahoo! Shopping Auctions

จากรูปที่ 2.6 คือโครงสร้างเนื้อหาของเว็บ ซึ่งจะเห็นว่าในเว็บเพจประกอบด้วยแต่ละบล็อกคือ VB1, VB2, VB3 และ VB4 โดยใน VB1 นั้นก็จะสามารถแบ่งได้เป็น VB1_1 และ VB1_2 และใน VB2 ก็เช่นกัน สามารถแบ่งได้เป็น VB2_1, VB2_2, VB2_3 ขณะเดียวกันบล็อกดังกล่าวยังสามารถแบ่งออกไปได้อีก เช่นใน VB2_1 นั้นสามารถแบ่งออกได้เป็น VB2_1_1 และ VB2_1_2 เป็นต้น

2.3.2 ขั้นตอนการทำงานของอัลกอริทึมวีไอพีเอส

จากรูปที่ 2.7 นั้นแสดงขั้นตอนการทำงานของ อัลกอริทึมวีไอพีเอส โดยจะเริ่มจากการแบ่งเป็นบล็อกจาก เอชทีเอ็มแอล ดอมทรีและข้อมูลที่แสดงในหน้าเว็บ หลังจากนั้นก็จะทำการหาตัวแบ่งระหว่างบล็อกที่ได้จากการแบ่งเอชทีเอ็มแอล ดอมทรี ในขั้นตอนแรกและให้ค่าน้ำหนักแต่ละตัวแบ่งและขั้นตอนสุดท้ายจะเป็นการสร้างโครงสร้างเนื้อหา จากนั้นจะเป็นการตรวจสอบว่าตรงตามความต้องการหรือไม่ ถ้าไม่ก็จะทำซ้ำขั้นตอนแรกอีกครั้งแต่ถ้าตรงตามความต้องการแล้วก็จะได้โครงสร้างเนื้อหาขึ้นมา



รูปที่ 2.7 ขั้นตอนการทำงานของอัลกอริทึมวีไอพีเอส

โดยขั้นตอนต่างๆสามารถอธิบายได้ดังต่อไปนี้

1. **การตัดออกเป็นบล็อก (Visual Block Extraction)**

เป็นการหาส่วนย่อยๆซึ่งจะเรียกว่าบล็อกของเว็บเพจโดยใช้กฎการแบ่งเพจออกเป็นบล็อก ซึ่งพิจารณาจากดอมทรีและมีการตั้งค่าดีโอซี (Degree of Coherence : DoC) เพื่อกำหนดว่าจะให้แต่ละบล็อกเป็นกลุ่มอย่างไร ซึ่งค่าดีโอซีนั้นมีคุณสมบัติดังนี้ คือยิ่งค่าดีโอซี สูงมาก แสดงว่าเนื้อหาภายในบล็อกนั้นมีความสอดคล้องกันมากขึ้นและในโครงสร้างลำดับชั้นของต้นไม้ ค่าดีโอซีของโหนดลูกต้องมากกว่าโหนดพ่อแม่

ตารางที่ 2.3 กฎในการแบ่งเว็บเพจออกเป็นบล็อก

กฎ	คำอธิบาย
R1	ถ้าดอมโหนดไม่ใช่ โหนดข้อความ (Text Node) และไม่มีโหนดลูกที่สามารถมองเห็นผ่านเว็บบราวเซอร์ (Valid Node) ได้ก็จะไม่ทำการแบ่งโหนดออกเป็นโหนดย่อยอีก
R2	ถ้าดอมโหนดมีแค่หนึ่งโหนดลูก (Child Node) ที่สามารถมองเห็นผ่านเว็บบราวเซอร์และไม่ได้เป็นโหนดข้อความให้แบ่งโหนด
R3	ถ้าดอมโหนดเป็นโหนดราก (Root Node) และมีแค่หนึ่งดอมทรีย่อยให้แบ่งโหนดได้
R4	ถ้าโหนดลูกทั้งหมดของดอมโหนดเป็นโหนดข้อความหรือโหนดที่แสดงคุณสมบัติเกี่ยวกับข้อความที่มีโหนดลูกเป็นข้อความ (Virtual Text Node) จะไม่ทำการแบ่งโหนด - ถ้าขนาดตัวอักษรและความหนาของตัวอักษรของโหนดลูกทุกโหนดเท่ากันให้กำหนดค่าดีโอซี = 10 - ถ้าไม่เท่ากันให้กำหนดค่าดีโอซี = 9
R5	ถ้ามีหนึ่งโหนดลูกของดอมโหนดเป็นโหนดที่ไม่ได้แสดงคุณสมบัติเกี่ยวกับข้อความ (Line-Break Node) ให้ทำการแบ่งโหนด
R6	ถ้ามีหนึ่งโหนดลูกของดอมโหนดมีแท็ก <HR> ให้ทำการแบ่งโหนด
R7	ถ้าขนาดของผลรวมของโหนดลูกมากกว่า ขนาดของดอมโหนด ให้ทำการแบ่งโหนด
R8	ถ้าสีพื้นหลังของดอมโหนดมีโหนดลูกหนึ่งโหนดที่มีสีพื้นหลังต่างกันให้ทำการแบ่งโหนด แต่จะยังไม่ทำการแบ่งโหนดลูกในขณะนี้ - กำหนดค่าดีโอซีมีค่าเท่ากับ 6-8 โดยดูจากแท็กเอชทีเอ็มแอลและขนาดของโหนดลูก
R9	ถ้าดอมโหนดมีอย่างน้อยหนึ่งโหนดลูกที่เป็นข้อความหรือโหนดที่ใช้แสดงคุณสมบัติเกี่ยวกับข้อความที่มีโหนดลูกเป็นข้อความและมีขนาดน้อยกว่าขนาดที่กำหนด ไม่ทำการแบ่งโหนด - กำหนดให้ค่าดีโอซีมีค่าเท่ากับ 5-8 ตามแท็กเอชทีเอ็มแอลของโหนดลูก
R10	ถ้าโหนดลูกของโหนดที่มีขนาดใหญ่ที่สุดนั้นมีขนาดเล็กกว่าขนาดที่กำหนด ไม่ทำการแบ่งโหนด
R11	ถ้าโหนดพี่น้อง (sibling node) ก่อนหน้าไม่ถูกแบ่งก็จะไม่ทำการแบ่งโหนด
R12	ทำการแบ่งโหนด
R13	ไม่ทำการแบ่งโหนด - กำหนดให้ค่าดีโอซีตามแท็กเอชทีเอ็มแอล

โดยค่าดีโอซีนั้นจะเป็นค่าจำนวนเต็มตั้งแต่ 1-10 นอกจากนี้ เราต้องกำหนดค่าพีดีโอซี (Permitted Degree of Coherence : PDoC) ก่อนเพื่อให้ได้โครงสร้างเนื้อหาที่มีความแตกต่างกัน โดยหากค่าพี

ดีโอซี น้อยจำนวนบล็อกที่ได้ในขั้นตอนสุดท้ายก็จะน้อยตามไปด้วย จึงเหมือนกับว่าค่าพีดีโอซีเป็นการกำหนดจำนวนบล็อก

สิ่งที่เราพิจารณาในการแบ่งดอมทรี มีดังนี้คือ

- พิจารณาที่ตัวดอมโหนด ตัวอย่างเช่น แท็กเอชทีเอ็มแอล, สีพื้นหลัง , ขนาดและรูปร่างของดอมโหนดดังกล่าว
- พิจาณาที่ตัวโหนดลูกของดอมโหนด ตัวอย่างเช่น แท็กเอชทีเอ็มแอล, สีพื้นหลัง , ขนาดและรูปร่างของโหนดลูก

สิ่งที่ใช้ในการพิจารณาในการสร้างกฎการแบ่งเว็บเพจออกเป็นบล็อกโดยดูจากดังนี้คือ

- แท็ก : เช่นแท็ก <hr> จะถูกใช้ในการแบ่งหัวข้อที่ต่างกันออกจากกัน
- สี : พิจารณาที่สีของแบกกราวด์เช่นหากสีพื้นหลังต่างกันก็จะถูกแบ่ง
- ข้อความ : พิจารณาที่ข้อความในแท็กนั้น เช่น หากเป็นข้อความทั้งหมดเราก็จะไม่ทำการแบ่ง
- ขนาด : พิจารณาที่ขนาดโดยเทียบจากขนาดทั้งหมดของเพจ เช่น หากขนาดของโหนดที่เราพิจารณานั้นมีขนาดเล็กเกินกว่าที่กำหนดเราก็จะไม่ทำการแบ่ง

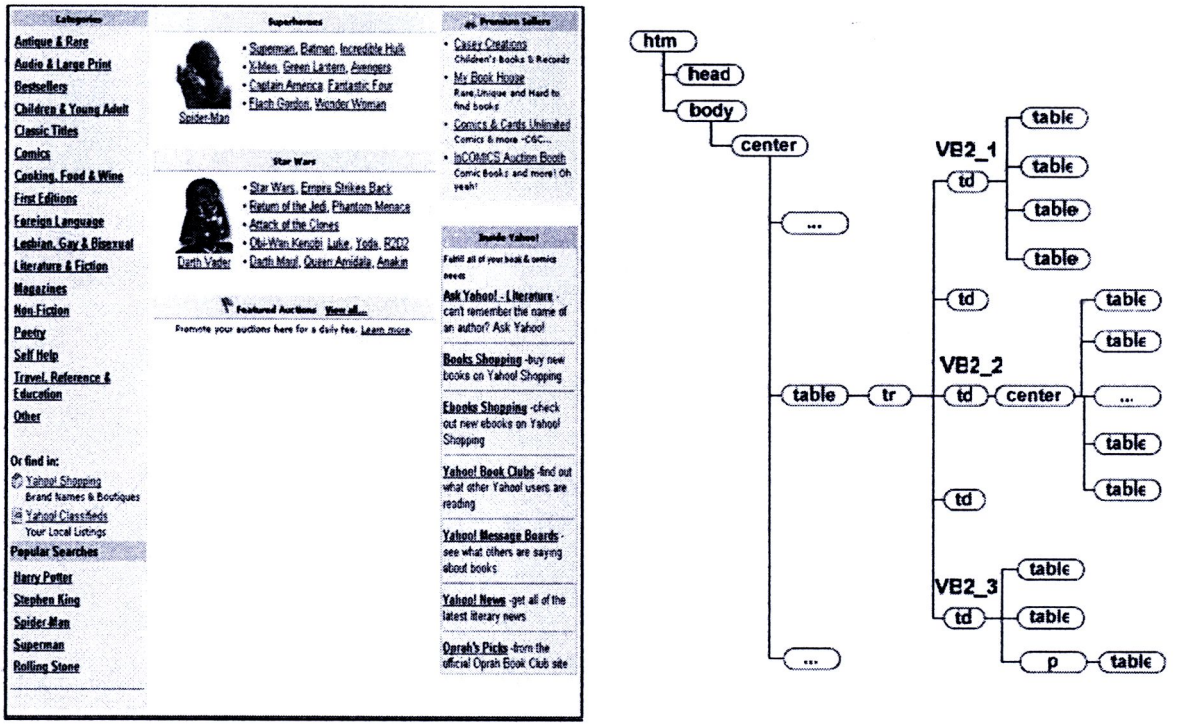
จากตารางที่ 2.3 กฎที่ได้เกิดจากการพิจารณา แท็ก, สี, ข้อความ, ขนาด จากนั้นเราจะทำการพิจารณาแต่ละโหนดของดอมทรีตามกฎว่าจะสามารถทำการแบ่งแต่ละโหนดนั้นได้หรือไม่ ซึ่งหากโหนดนั้นไม่ได้ทำการแบ่งก็จะมีการเซตค่าดีโอซีให้ในแต่ละโหนดด้วย

ตารางที่ 2.4 กฎที่แตกต่างกันของแท็กที่แตกต่างกัน

แท็ก	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13
<table>	/	/	/					/		/			/
<tr>	/	/	/				/	/		/			/
<td>	/	/	/	/					/	/	/		/
<p>	/	/	/	/	/	/	/		/	/		/	
แท็กที่แสดงคุณสมบัติเกี่ยวกับข้อความ	/	/	/	/	/	/	/		/	/		/	
แท็กอื่นๆ	/	/	/	/		/	/		/	/		/	

จากตารางที่ 2.4 จะแสดงว่าในแท็กที่ต่างกันก็จะมีการใช้กฎที่ต่างกัน ยกตัวอย่างเช่น <table> จะใช้เฉพาะกฎข้อ 1, 2, 3, 8, 9, 10, 13 เท่านั้น เป็นต้น ซึ่งความหมายของแต่ละแท็กสามารถบอกได้ดังนี้ คือ <table> ใช้แสดงตาราง, <tr> ใช้ขึ้นแถวใหม่, <td> ขึ้นคอลัมน์ใหม่, <p> กำหนดเริ่มต้นของย่อหน้า , แท็กที่แสดงคุณสมบัติเกี่ยวกับข้อความเช่น <a>, <acrunym>, <abbr>, <b>, <big>, <cite>, <code>, <del>, <dfn>, ,<font>, <i>, <img>, <input>, <ins>, <nobr>, <kbd>, <q>, , <sub>, <sup>, <u>, <var>, <samp>, <small>,

จากรูปที่ 2.8 ส่วนที่แสดงต่อไปนี้ เป็นส่วนที่มาจากแท็กตารางซึ่งเป็นส่วนหลักของหน้าเว็บ โดยดอมทรีแสดงดังทางขวาในขั้นตอนการแบ่งบล็อกนั้นเมื่อเราเจอ <table> ซึ่งมีโหนดลูกแค่โหนดเดียวคือ <tr> ซึ่งก็เป็นไปตามกฎข้อ 2 เราจึงทำการดูต่อในโหนด <tr> ซึ่งมี <td> 5 โหนด แต่มีเพียง 3 โหนดที่เป็นโหนดที่มองเห็นผ่านเว็บบราวเซอร์ซึ่ง <td> โหนดแรกนั้นมีสีพื้นหลังที่ต่างจากโหนดพ่อแม่ของ จากกฎข้อที่ 8 จึงทำการแยก <td> อันแรกออกมาได้หนึ่งบล็อกและจะไม่แบ่งต่อ หลังจากนั้นตรวจสอบ <td> ตัวที่ 2 ซึ่งเป็นโหนดที่ไม่สามารถมองเห็นผ่านเว็บบราวเซอร์ (Invalid Node) จึงทำการตัดออก และจากกฎข้อ 11 ถ้าโหนดพี่น้องก่อนหน้าไม่ได้ทำการแบ่งโหนดต่อมาก็จะไม่ทำการแบ่งด้วย ดังนั้น <td> ที่ 3 และ 5 จึงยังไม่ถูกแบ่งในรอบนี้



รูปที่ 2.8 แสดงตัวอย่างการตัดออกเป็นบล็อกของตัวอย่างเว็บ

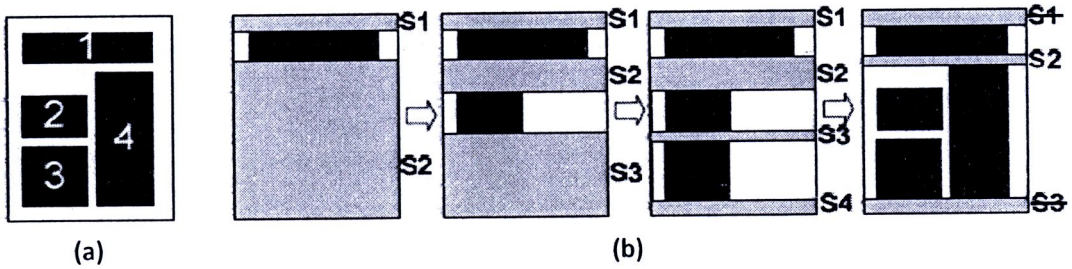
2. การหาตัวแบ่ง (Visual separator detection)

หลังจากที่แบ่งเว็บเพจเป็นส่วนย่อย ๆได้เป็นบล็อกแล้วจะทำการหาตัวแบ่งแยกเพื่อแบ่งคุณลักษณะที่แตกต่างกันออกจากกัน โดยตัวแบ่งนั้นอาจจะเป็นแนวตั้งหรือแนวนอนก็ได้ โดยตัวแบ่งนั้นถูกแสดงด้วย 2-tuple (P_s,P_e) P_s คือ จุดเริ่มต้น P_e คือจุดสิ้นสุดและความกว้างของตัวแบ่งจะคำนวณได้จากความกว้างของสองตัวแปรนี้ เมื่อทำการหาตัวแบ่งระหว่างแต่ละบล็อกแล้ว ก็จะทำการให้ค่าน้ำหนักแต่ละตัวแบ่งเพื่อนำไปสร้างเป็นโครงสร้างเนื้อหาต่อไป โดยขั้นตอนการหาตัวแบ่งมีดังนี้คือ

(1) การหาตัวแบ่ง

การหาตัวแบ่งสามารถอธิบายได้ดังต่อไปนี้ คือ

1. ตัวแบ่งแรกจะเริ่มจากขอบของหน้าเว็บก่อน โดยจะเริ่มจากตัวแบ่งเดียวคือ (P_{be},P_{ee})
2. สำหรับทุกบล็อก แต่ละตัวแบ่งจะถูกกำหนดได้ดังนี้คือ
   - ถ้ามีบล็อกอยู่ตัดตัวแบ่ง ให้เพิ่มตัวแบ่ง
   - ถ้ามีบล็อกอยู่คลอบคลุมตัวแบ่ง ให้เอาตัวแบ่งออก
3. หลังจากที่ได้ตัวแบ่งทั้งหมดเอาตัวแบ่งเฉพาะส่วนที่อยู่ตรงขอบทั้งสี่ของหน้าเว็บออกแสดงได้ดังตัวอย่างต่อไปนี้



รูปที่ 2.9 แสดงตัวอย่างการหาขั้นตอนการหาตัวแบ่ง

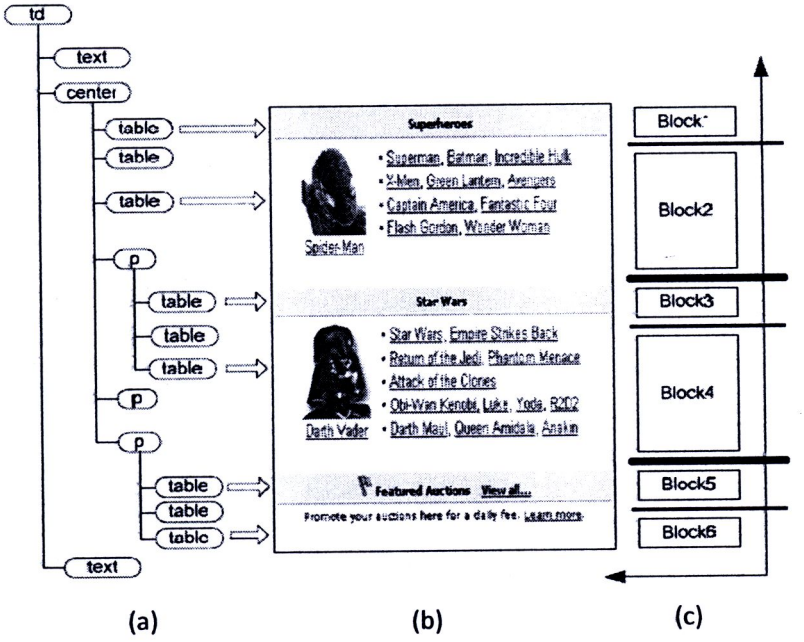
จากรูปที่ 2.9 จะแสดงตัวอย่างการขั้นตอนหาตัวแบ่งโดยจะแสดงเฉพาะแนวนอนเท่านั้น ซึ่งเห็นว่า หลังจากเรานำดอมทรีของเว็บเพจมาแบ่งจะได้บล็อกออกมาดังรูปที่ 2.9 (a) จากนั้นการหาตัวแบ่งจะเริ่มจากตัวแบ่งแรกคือ S_1 ซึ่งอยู่ขอบของหน้าเว็บก่อน หลังจากที่บล็อก ที่ 1 ถูกนำเข้าไปก็จะทำการแยกตัวแบ่งได้ออกเป็น S_1 กับ S_2 จากนั้นนำบล็อกที่ 2 เข้าไปจะได้ตัวแบ่ง S_3 และนำบล็อกที่ 3 ใส่เข้าไปก็จะได้ตัวแบ่ง S_4 ออกมาเช่นกัน สุดท้ายบล็อกที่ 4 เมื่อนำเข้าไปหาตัวแบ่งจะเห็นว่าบล็อกที่ 4 นั้นอยู่อยู่ตัดทับตัวแบ่ง S_2 อยู่และคลอบคลุมตัวแบ่ง S_3 ด้วยก็จะทำการเพิ่มตัแบ่ง S_2 และตัวแบ่ง S_3 ก็จะถูกเอาออก หลังจากหาตัวแบ่งได้แล้วจะได้ตัวแบ่งทั้งหมดคือ S_1, S_2 และ S_3 และขั้นตอนสุดท้ายตัวแบ่งที่อยู่ตรงขอบคือตัวแบ่ง S_1 และ S_3 ก็จะถูกตัดออกเหลือเพียง S_2 ดังรูป 2.9 (b)

(2) การให้ค่าน้ำหนักแก่ตัวแบ่ง

หลังจากที่ได้ตัวแบ่งระหว่างแต่ละบล็อกจากขั้นตอนการหาตัวแบ่งออกมาแล้วก็จะทำการให้ค่าน้ำหนักแต่ละตัวแบ่งเพื่อใช้ในการแบ่งบล็อกที่มีความหมายแตกต่างกันออกจากกันโดยดูจากบล็อกที่อยู่ข้างเคียง ซึ่งมีกฎในการให้ค่าน้ำหนักแต่ละตัวแบ่งดังนี้

- ยิ่งมีระยะทางระหว่างบล็อกในแต่ละข้างของตัวแบ่งมาก จะมีน้ำหนักมาก
- ถ้าตัวแบ่งนั้นมีการซ้อนทับกันกับบางแท็กเอชทีเอ็มแอล เช่น แท็ก <HR> จะมีน้ำหนักมาก
- ถ้าสีของพื้นหลังของทั้งสองข้างของตัวแบ่งแตกต่างกัน จะมีค่าน้ำหนักมาก
- สำหรับตัวแบ่งแนวนอน หากคุณสมบัติของตัวอักษร เช่น ขนาดนั้นใหญ่กว่าสองข้างของตัวแบ่ง ค่าน้ำหนักจะเพิ่มขึ้น นอกจากนี้หากขนาดตัวหนังสือที่อยู่ข้างบนของตัวแบ่งนั้นมีขนาดใหญ่กว่าตัวแบ่งข้างล่าง ค่าน้ำหนักจะเพิ่มขึ้นเช่นกัน

- สำหรับตัวแบ่งแนวนอนหากโครงสร้างระหว่างสองข้างของตัวแบ่งนั้นคล้ายๆ กัน ค่าน้ำหนักจะลดลง



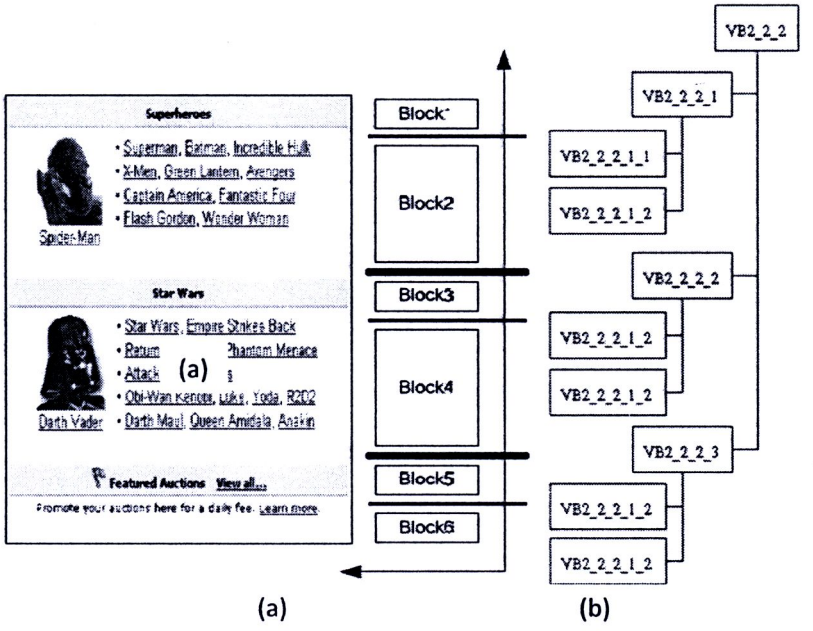
รูปที่ 2.10 (a) แสดงดอมทรี (b) แสดงส่วนย่อยของหน้าเพจ
(c) แสดงตัวแบ่งและค่าน้ำหนักระหว่างแต่ละบล็อก

จากรูปที่ 2.10 จะแสดงตัวอย่างการให้ค่าน้ำหนักของแต่ละตัวแบ่ง โดยจากรูป 2.10 (b) ได้นำบางส่วนจากเพจซึ่งคือส่วนของ <td> ที่ 3 ในรูปที่ 2.8 มาแสดง ซึ่งในรูปที่ 2.10 (a) นี้ จะแสดงโครงสร้างของดอมทรี โดยจะเห็นว่าในดอมทรีนั้นจะมีหลายโหนดที่เป็นโหนดที่ไม่สามารถมองเห็นผ่านเว็บบราวเซอร์ซึ่งก็จะไม่ถูกแบ่งออกเป็นบล็อก ในขั้นตอนแรกคือการตัดออกเป็นบล็อกสามารถแบ่งได้เป็น 6 บล็อก และ 5 ตัวแบ่งแนวนอน หลังจากนั้นตัวแบ่งแต่ละตัวก็จะถูกให้ค่าน้ำหนักตามกฎ 5 ข้อ จากตัวอย่างนี้ ตัวแบ่งที่ 2 ซึ่งก็คือตัวแบ่งระหว่างบล็อกที่ 2 และบล็อกที่ 3 นั้นจะมีค่าน้ำหนักมากกว่าตัวแบ่งระหว่าง บล็อกที่ 1 และบล็อก ที่ 2 เพราะความแตกต่างของขนาดและน้ำหนักของตัวอักษรและตัวแบ่งระหว่าง บล็อกที่ 4 และบล็อกที่ 5 จะมีค่าน้ำหนักมากเพราะความแตกต่างของขนาดและน้ำหนักของตัวอักษร เช่นเดียวกัน โดยจากรูปที่ 2.10 (c) จะเห็นว่าตัวแบ่งที่มีความหนาจะแสดงว่าตัวแบ่งนั้นมีค่าน้ำหนักมาก

3. การสร้างโครงสร้างเนื้อหา (Content Structure Construction)

หลังจากที่ได้ทำการหาตัวแบ่งและให้ค่าน้ำหนักตัวแบ่งแล้ว ก็จะเป็นการสร้างโครงสร้างเนื้อหาขึ้นมาโดยในขั้นตอนนี้จะทำการพิจารณาตัวแบ่งที่ได้ให้ค่าน้ำหนักว่าแต่ละบล็อกที่ถูกแบ่งไว้ตั้งแต่ขั้นตอนแรกจะถูกรวมกันได้อย่างไร ซึ่งขั้นตอนการสร้างโครงสร้างเนื้อหา จะเริ่มจากการพิจารณาตัวแบ่งที่มีค่าน้ำหนักน้อยก่อนและบล็อกที่อยู่ระหว่างตัวแบ่งนี้ก็จะถูกรวมเป็นบล็อกใหม่ ซึ่ง

ขั้นตอนการรวมนั้นจะถูกทำไปเรื่อยๆจนกว่าจะเจอตัวแบ่งที่มีค่าน้ำหนักมาก จากนั้นค่าดีโอซีของบล็อกใหม่จะถูกตั้งค่าขึ้นตามตัวแบ่งที่มีค่าน้ำหนักมาก หลังจากนั้นแต่ละโหนดปลาย (Leaf Node) ก็จะถูกตรวจสอบว่าตรงตามความต้องการหรือไม่ ถ้าไม่ก็จะเข้าสู่ขั้นตอนการตัดออกเป็นบล็อกอีกครั้งเพื่อทำการแบ่งโหนดปลายนั้นต่อไปแต่ถ้าทุกโหนดตรงตามความต้องการแล้ว ก็จะไม่ทำการแบ่งต่อและจะได้โครงสร้างเนื้อหา สำหรับเว็บเพจนี้ขึ้นมา โดยการที่จะดูว่าตรงตามความต้องการหรือไม่จะดูจากค่าดีโอซี หากค่าดีโอซีมากกว่าพีดีโอซี แสดงว่าตรงตามความต้องการโดยที่ค่าพีดีโอซีคือค่าที่เรากำหนดไว้ก่อน ซึ่งจะแสดงตัวอย่างการรวมเป็นบล็อกได้จากตัวอย่างต่อไปนี้



รูปที่ 2.11 แสดงตัวอย่างของการสร้างโครงสร้างเนื้อหา

จากรูปที่ 2.11 จะแสดงตัวอย่างการสร้างโครงสร้างเนื้อหาโดยจะเป็นการพิจารณาตัวแบ่งที่มีค่าน้ำหนักน้อยและบล็อกที่อยู่ระหว่างตัวแบ่งนั้นก็จะสามารถรวมกันได้ ดังรูป 2.11 (a) ก่อนอื่นตัวแบ่งที่ 1, 3, 5 ซึ่งมีค่าน้ำหนักน้อยจะถูกพิจารณาก่อนโดยบล็อกที่ 1 กับ 2 สามารถรวมกันได้เป็นบล็อกใหม่คือ VB2_2_2_1 เช่นเดียวกับบล็อกที่ 3 กับ 4 จะได้บล็อกใหม่คือ VB2_2_2_2 และสุดท้ายการรวมบล็อกที่ 5 กับ 6 จะได้ VB2_2_2_3 โดยบล็อกที่ได้ใหม่จากการดูค่าน้ำหนักจะมี 3 บล็อกคือ VB2_2_2_1 , VB2_2_2_2 , VB2_2_2_3 ซึ่งจะเห็นว่าเป็นลูกของ VB2_2_2 ทั้งสิ้น จากนั้นแต่ละโหนดปลาย เช่น VB2_2_2_1_1, VB2_2_2_1_2, VB2_2_2_2_2 เป็นต้น จะถูกตรวจสอบว่าตรงความต้องการหรือไม่ซึ่งถ้าตรงแล้วก็จะได้โครงสร้างเนื้อหา ขึ้นมาดังรูปที่ 2.11(b)