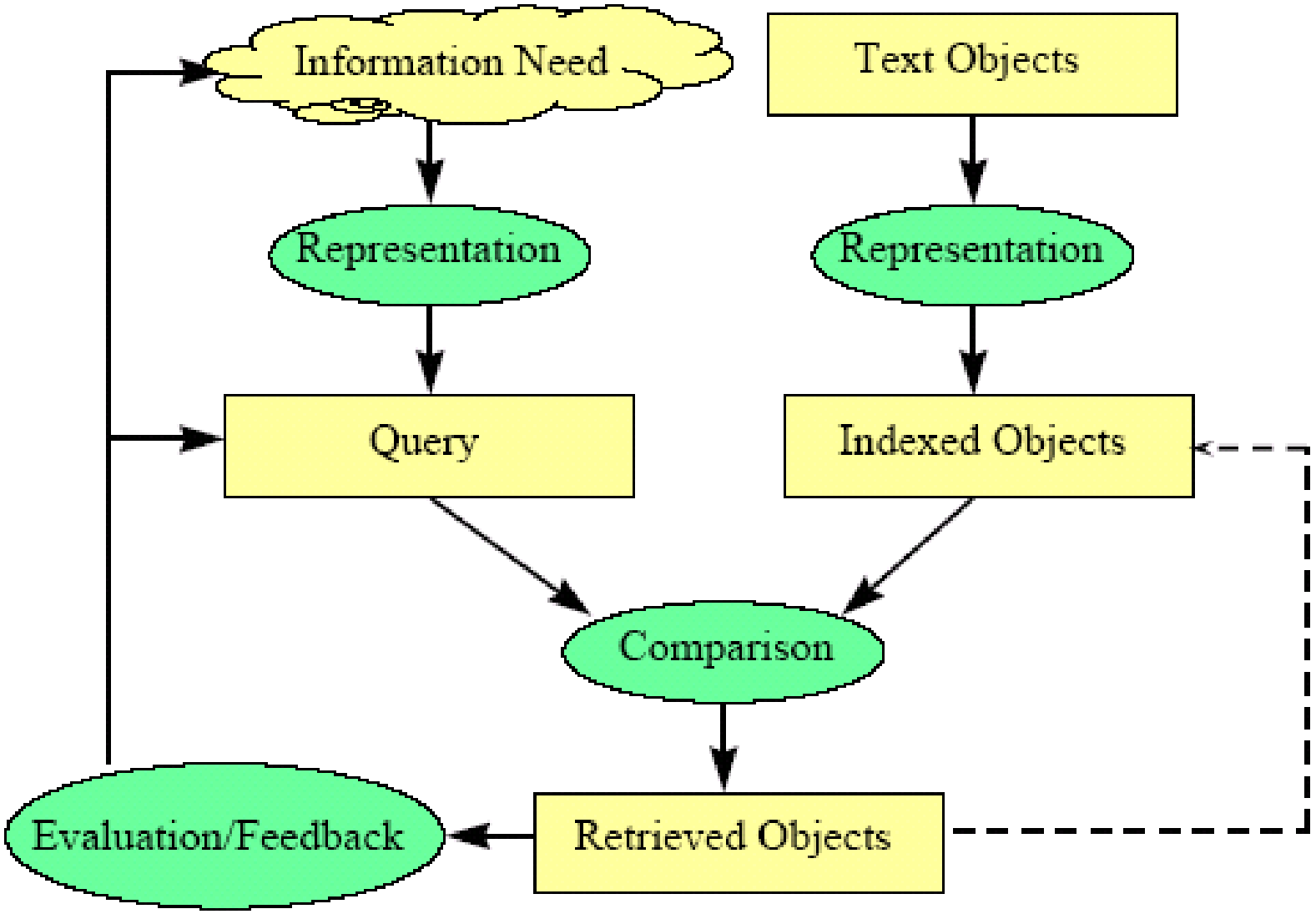
ครั้งที่ 8

กลยุทธ์การค้นหาข้อมูล

Simple model of IR



Comparision

- นำข้อคำถามมาเปรียบเทียบกับตัวแทนคลัสเตอร์
- สนใจเพียงแต่ว่า เอกสารที่ถูกดึงออกมานั้นเกี่ยวข้อง (Relevant) กับข้อคำถาม (Request)

กลยุทธ์การค้นหาข้อมูล

- **Boolean Search**

- **Serial Search**
  - **Matching Functions**

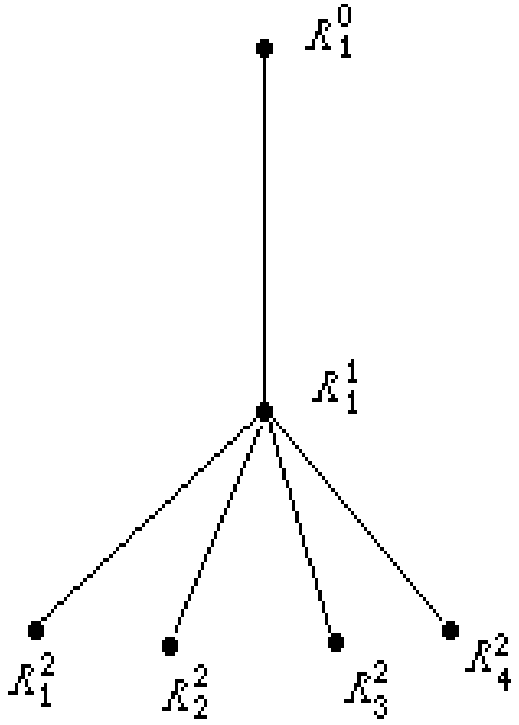
Boolean Search

- นิยมใช้มาก
- การสืบค้นจะดึงเอกสารที่ผลจากการเปรียบเทียบทางตรรกะระหว่างข้อคำถามกับเอกสารมีค่าเป็นจริง (True) เท่านั้น
- ข้อคำถามที่ดีจะถูกแสดงในรูปแบบของคำดัชนี และถูกเชื่อมโยงด้วยตัวกระทำทางตรรกะ AND,OR และ NOT

Q = (K1 AND K2) OR (K3 AND K4)

Boolean Search กำหนดขอบเขตในการค้นหาของผู้ใช้ให้อยู่ในวงที่กำหนด

- โดยกำหนดให้ผู้ใช้เข้าถึงข้อมูลเฉพาะที่มีการกำหนดให้เท่านั้น
- ซึ่งคีย์เวิร์ดที่เกี่ยวข้องซึ่งอาจสืบค้นข้อมูลที่มีความหมายกว้างกว่าหรือสืบค้นข้อมูลที่เจาะจงมากกว่าก็ได้ ตามตัวอย่างในโครงสร้างของต้นไม้ (tree)



Boolean Search โดยผ่าน Inverted File

- บันทึกลิสท์ (List) หนึ่งสำหรับแต่ละคีย์เวิร์ดและในแต่ละลิสท์จะมีแอดเดรส (หรือหมายเลย) ของเอกสารที่บรรจุคีย์เวิร์ดนั้นๆ

K1 – List : D1, D2, D3, D4
K2 – List : D1, D2
K3 – List : D1, D2, D3
K4 – List : D1

Boolean Search ก็โดยผ่าน Inverted File

- การที่จะสามารถดึงเอกสารออกมาได้ตามข้อคำถามหนึ่งนั้น เราจะทำการเชื่อมโยงทางตรรกศาสตร์บน Ki – List ยกตัวอย่าง

K1 – List : D1, D2, D3, D4
K2 – List : D1, D2
K3 – List : D1, D2, D3
K4 – List : D3

ถ้าข้อคำถาม Q = (K1 AND K2) OR (K3 AND K4)

ซึ่งผลลัพธ์ที่ได้ก็คือ Set { D1, D2, D3}

Boolean Search in SQL

```
(SELECT docID FROM InvertedFile
  WHERE word = "window"
 INTERSECT
 SELECT docID FROM InvertedFile
  WHERE word = "glass" OR word = "door")
EXCEPT
SELECT docID FROM InvertedFile
 WHERE word="Microsoft"
ORDER BY magic_rank()
```

Fully Boolean Search

- วิธีที่อนุญาตให้มี AND Logic แต่นำเอาจำนวนที่แท้จริงของค่าดัชนีที่ข้อคำถามตรงกับของเอกสารหนึ่งมาคิดด้วย ตัวเลขจำนวนนี้ เราเรียกว่าเป็น Co-ordination Level

K1 – List : D1, D2, D3, D4
K2 – List : D1, D2
K3 – List : D1, D2, D3

เราจะได้ Ranking ต่อไปนี้

Co-ordination Level	
3	D1, D2
2	D3
1	D4

Fully Boolean Search

- ผลลัพธ์นั้นเกิดจากการใช้วิธีการวัดความคล้ายคลึงโดยวิธี Simple Matching

  คำนวณจาก $|D \cap Q|$

  กล่าวคือ ขนาดของโอเวอร์แล็พระหว่าง D และ Q

  ซึ่งจะถูกแสดงขนาดเป็นชุดของคีย์เวิร์ด

  ก็คือ Simple Matching Coefficient

Matching Functions

- วัดความใกล้ชิดระหว่างข้อคำถามกับตัวแทนเอกสารหรือตัวแทนคลัสเตอร์
- ฟังก์ชั่นที่ง่ายที่สุดได้แก่ฟังก์ชั่นที่เกี่ยวข้องกับ Simple Matching Search Strategies ดังนี้

กำหนด M เป็น Matching Function

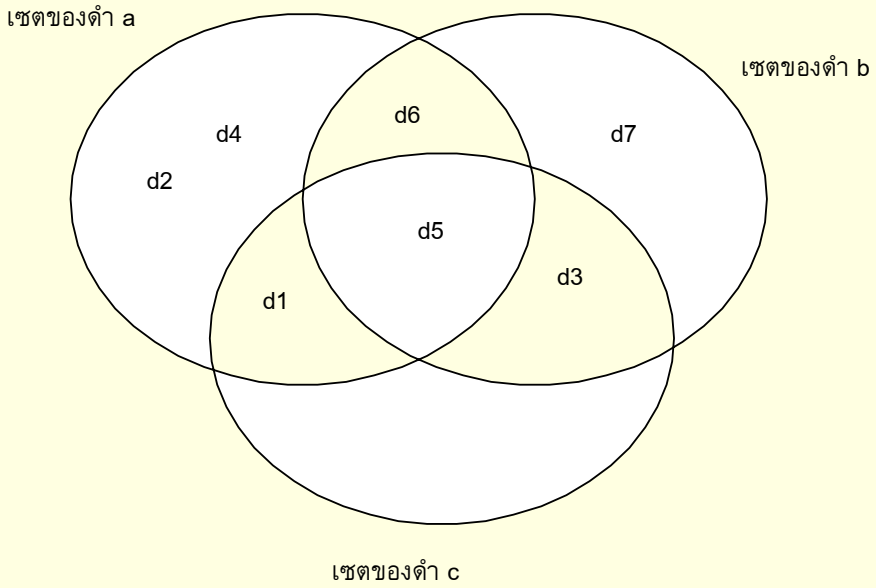
D เป็น เซ็ทของคีย์เวิร์ดที่ใช้แทนเอกสาร

Q เป็น เซ็ทของคีย์เวิร์ดที่ใช้แทน Query

ดังนั้น

$$M = \frac{2|D \cap Q|}{|D| + |Q|}$$

กลยุทธ์ในการค้นหาโดยใช้ Matching Function



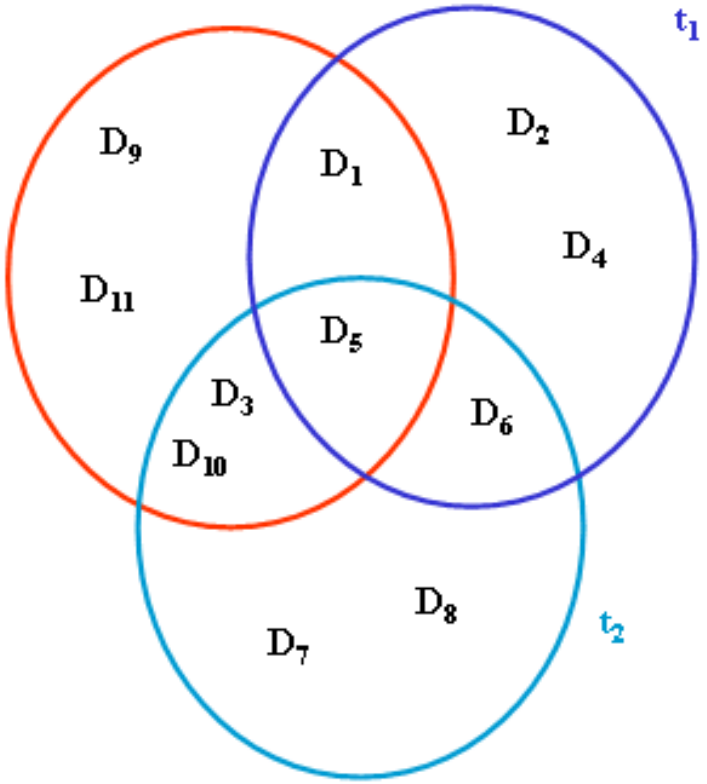
	a	b	c	q*dj
d1	1	0	1	2
d2	1	0	0	1
d3	0	1	1	2
d4	1	0	0	1
d5	1	1	1	3
d6	1	1	0	2
d7	0	1	0	1

q	1	1	1	

กลยุทธ์ในการค้นหาโดยใช้ Matching Function

docs	*t1*	*t2*	*t3*	RSV=Q.Di
D1	1	0	1	4
D2	1	0	0	1
D3	0	1	1	5
D4	1	0	0	1
D5	1	1	1	6
D6	1	1	0	3
D7	0	1	0	2
D8	0	1	0	2
D9	0	0	1	3
D10	0	1	1	5
D11	0	0	1	4
Q	1	2	3	
	q1	*q2*	*q3*	

Q is a query – also represented



Boolean term combinations

Matching Functions

Cosine Correlation

เอกสารและข้อคำถามถูกแสดงแทนโดยเวคเตอร์ที่เป็นตัวเลขใน t-Space

$Q = (q1, q2, \ldots\ldots\ldots\ldots, qt)$ และ

$D = (d1, d2, \ldots\ldots\ldots\ldots, dt)$

ซึ่ง qi และ di เป็นน้ำหนักที่เกี่ยวข้องกับคีย์เวิร์ด i

Cosine Correlation

$$\text{Cosine Correlation} = \frac{\sum_{i=1}^{t} q_i d_i}{\left(\sum_{i=1}^{t} (q_i)^2 \sum_{i=1}^{t} (d_i)^2\right)^{1/2}}$$

หรือถ้าจะเขียนนิยามสำหรับ Vector Space หนึ่งที่มี Euclidean Norm

$$r = \frac{(Q, D)}{\|Q\| \|D\|} = \text{cosine } \alpha$$

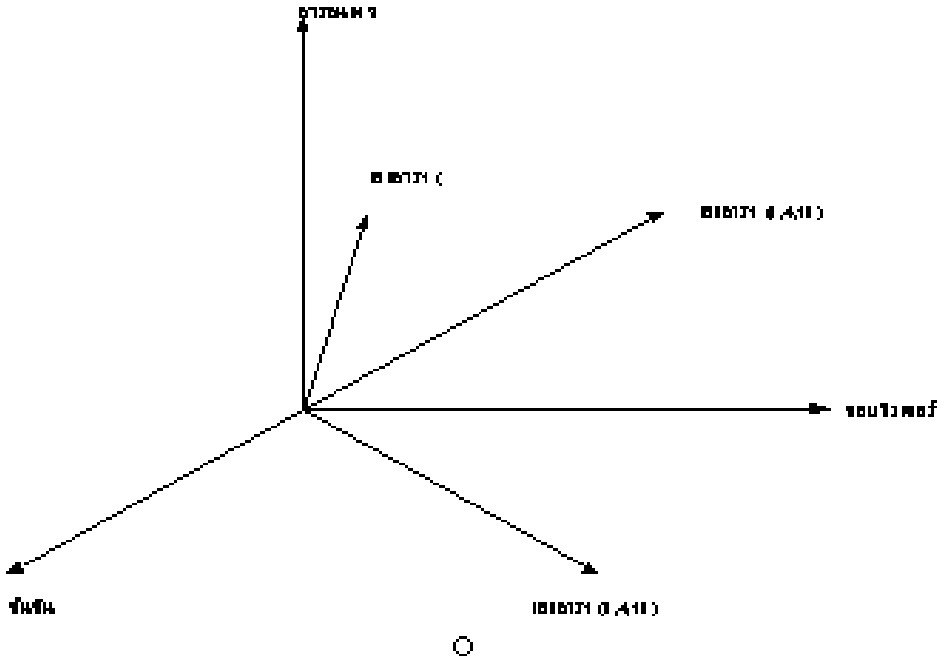
การแทนเอกสารด้วยเวกเตอร์โมเดล

ดำดัชนีในระบบประกอบด้วย สารสนเทศ ค้นคืน คอมพิวเตอร์

D1 = (1,0,1)

D2 = (0,0,1)

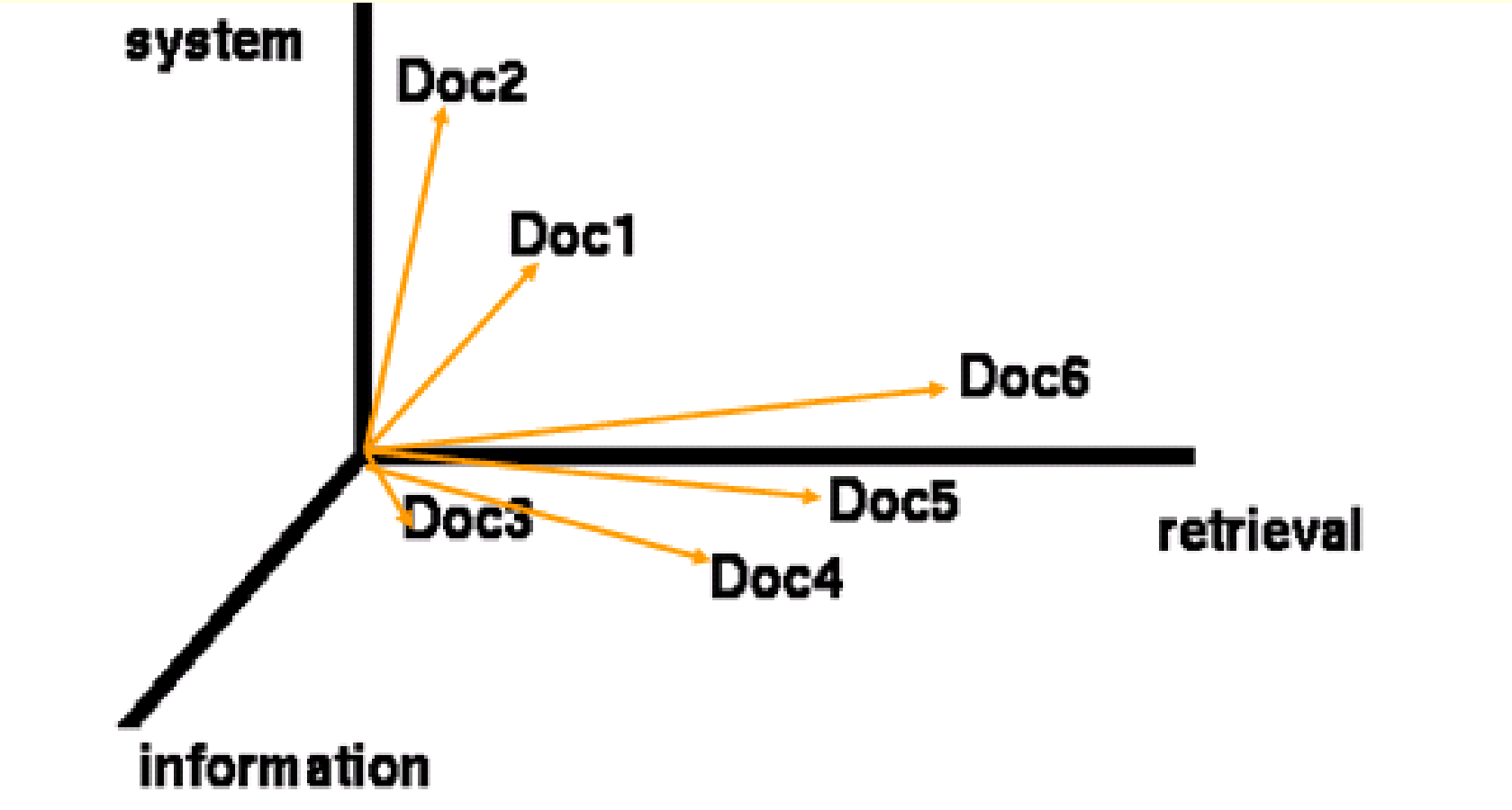
D3 = (1,1,1)



Vector Space Model:

- เป็นการนำเอกสารหรือข้อสอบถาม(queries)จะถูกนำมารวมด้วยกันใน space เป็นความคล้ายคลึงกัน
- น้ำหนักข้อสอบถามและเอกสาร เป็นพื้นฐานของความยาวและทิศทางของเวกเตอร์
- การวัดระยะทางของเวกเตอร์ระหว่างข้อสอบถามและเอกสารถูกใช้สำหรับเป็น rank ในการดึงเอกสาร

Vector Space Model:



การแทนเอกสารด้วยเวกเตอร์โมเดล

- ค่าความเหมือนระหว่างเอกสารและคำขอ คำนวณด้วยค่าของมุม cosine
- ผลลัพธ์ที่ได้สามารถจัดลำดับให้ตรงกับความต้องการของผู้ใช้ได้ โดยใช้หลักการในการถ่วงน้ำหนักโดยการจัดหมู่ข้อมูล (Clustering)
- การจัดหมู่ข้อมูลเป็นการแยกแยะว่าเอกสารใดตรงกับคำขอ เพื่อจำแนกว่าเอกสารใดบ้างควรอยู่ในเซต

นิยามของเวกเตอร์โมเดล

- กำหนดให้ wij เป็นนำหนักของคำที่ ki ของเอกสาร dj เอกสารแต่ละฉบับจะแทนด้วย dj = (w1j, w2j,…, wtj)
- คำค้นคืน (Query) ก็มีการให้นำหนักของคำด้วยแบบเดียวกัน คือ wiq เป็นนำหนักของคำที่ ki กับคำขอ q ดังนั้นเวคเตอร์คำขอจะอยู่ในรูปของ q = (w1q, w2q, …, wtq)
- การค้นคืนข้อมูลจะวัดด้วยค่าความคล้ายกันของเอกสารคือ

$$sim(d_j, q) = \frac{\bar{d_j} * \bar{q}}{|\bar{d_j}| * |\bar{q}|}$$

$$\frac{\sum_{i=1}^{t} w_{i,j} * w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} * \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

ตัวอย่าง

สมมุติว่าเอกสารและข้อสอบถามถูกแทนในรูปของเวกเตอร์

ตำแหน่งที่ 1 สอดคล้องกับ term 1

ตำแหน่งที่ 2 สอดคล้องกับ term 2

ตำแหน่งที่ t สอดคล้องกับ term t

น้ำหนักของแต่ละเทอมถูกเก็บในแต่ละตำแหน่ง

ดังนั้น

$$D_i = w_{d_{i1}}, w_{d_{i2}}, ..., w_{d_{it}}$$

$$Q = w_{q1}, w_{q2}, ..., w_{qt}$$

$$w = 0 \text{ if a term is absent}$$

ตัวอย่าง

สมมุติว่า เวกเตอร์ของเอกสารเป็นดังนี้

ID	nova	galaxy	heat	h'wood	film	role	diet	fur
A	10	5	3					
B	5	10						
C				10	8	7		
D				9	10	5		
E							10	10
F							9	10
G	5		7			9		
H		6	10	2	8			
I				7	5		1	3

ตัวอย่าง

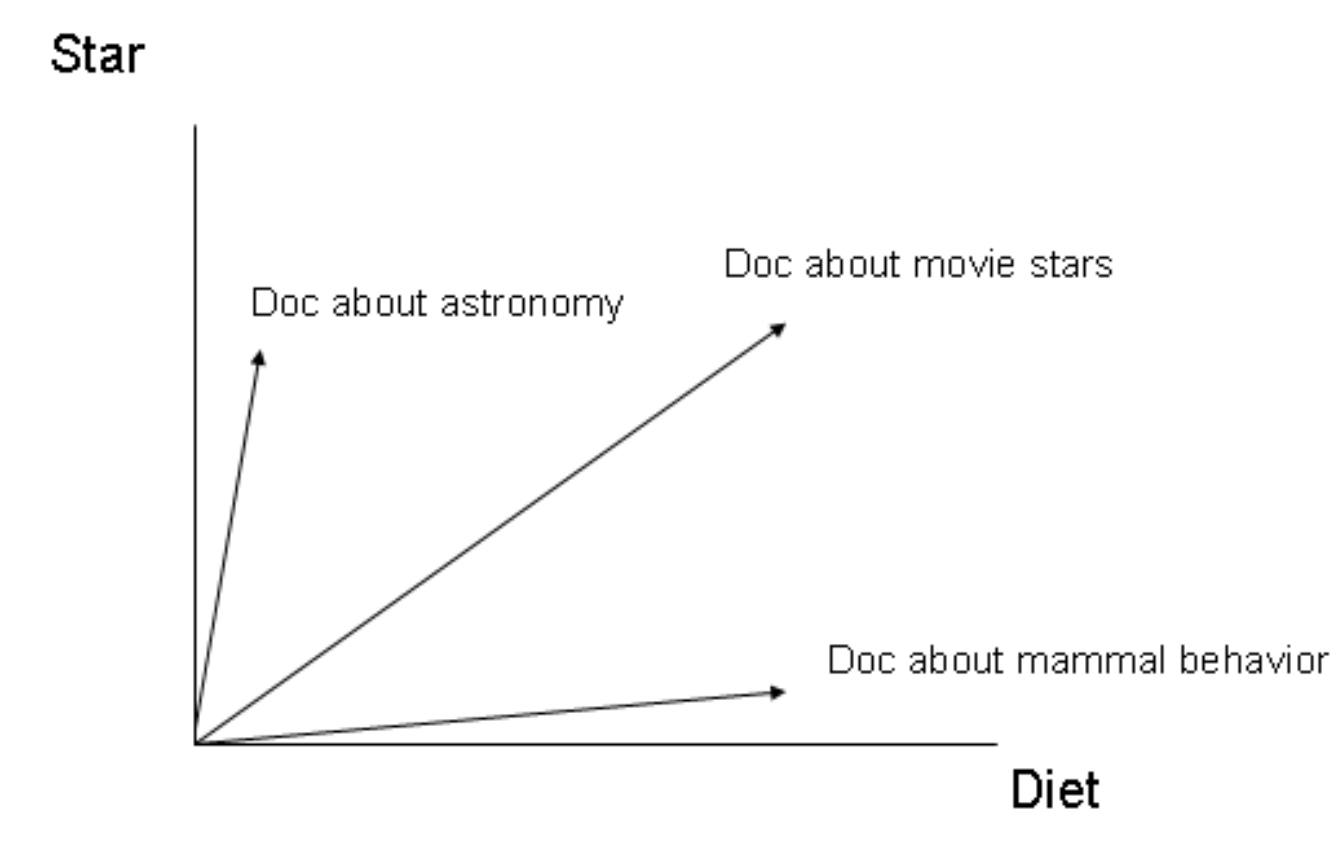
ID	nova	galaxy	heat	h'wood	film	role	diet	fur
A	10	5	3					
B	5	10						
C				10	8	7		
D				9	10	5		
E							10	10
F							9	10
G	5		7			9		
H		6	10	2	8			
I				7	5		1	3

จากข้อมูลข้างต้นเห็นได้ว่า

“Nova” occurs 10 times in text A
“Galaxy” occurs 5 times in text A
“Heat” occurs 3 times in text A
(Blank means 0 occurrences.)
“Hollywood” occurs 7 times in text I
“Film” occurs 5 times in text I
“Diet” occurs 1 time in text I
“Fur” occurs 3 times in text I

ตัวอย่าง

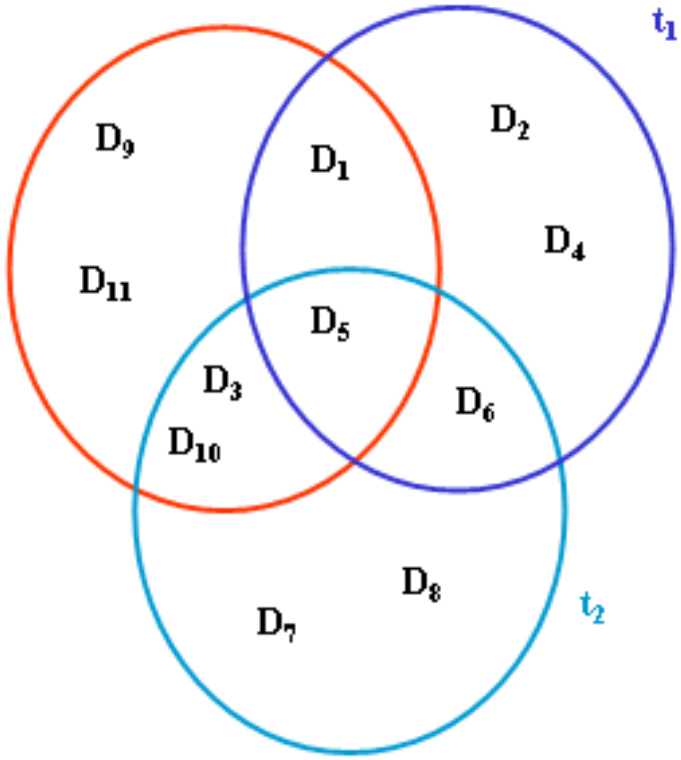
- เราสามารถ plot เวกเตอร์ได้ดังนี้



ตัวอย่าง

docs	*t1*	*t2*	*t3*	RSV=Q.Di
D1	1	0	1	4
D2	1	0	0	1
D3	0	1	1	5
D4	1	0	0	1
D5	1	1	1	6
D6	1	1	0	3
D7	0	1	0	2
D8	0	1	0	2
D9	0	0	1	3
D10	0	1	1	5
D11	0	0	1	4
Q	1	2	3	
	q1	*q2*	*q3*	

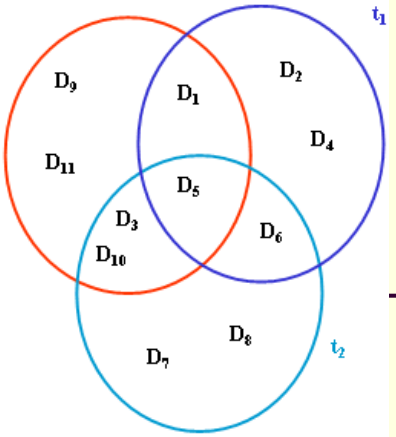
Q is a query – also represented as a vector



Boolean term combinations

docs	t1	t2	t3	RSV=Q.Di
D1	1	0	1	4
D2	1	0	0	1
D3	0	1	1	5
D4	1	0	0	1
D5	1	1	1	6
D6	1	1	0	3
D7	0	1	0	2
D8	0	1	0	2
D9	0	0	1	3
D10	0	1	1	5
D11	0	0	1	4
Q	1	2	3	
	q1	q2	q3	

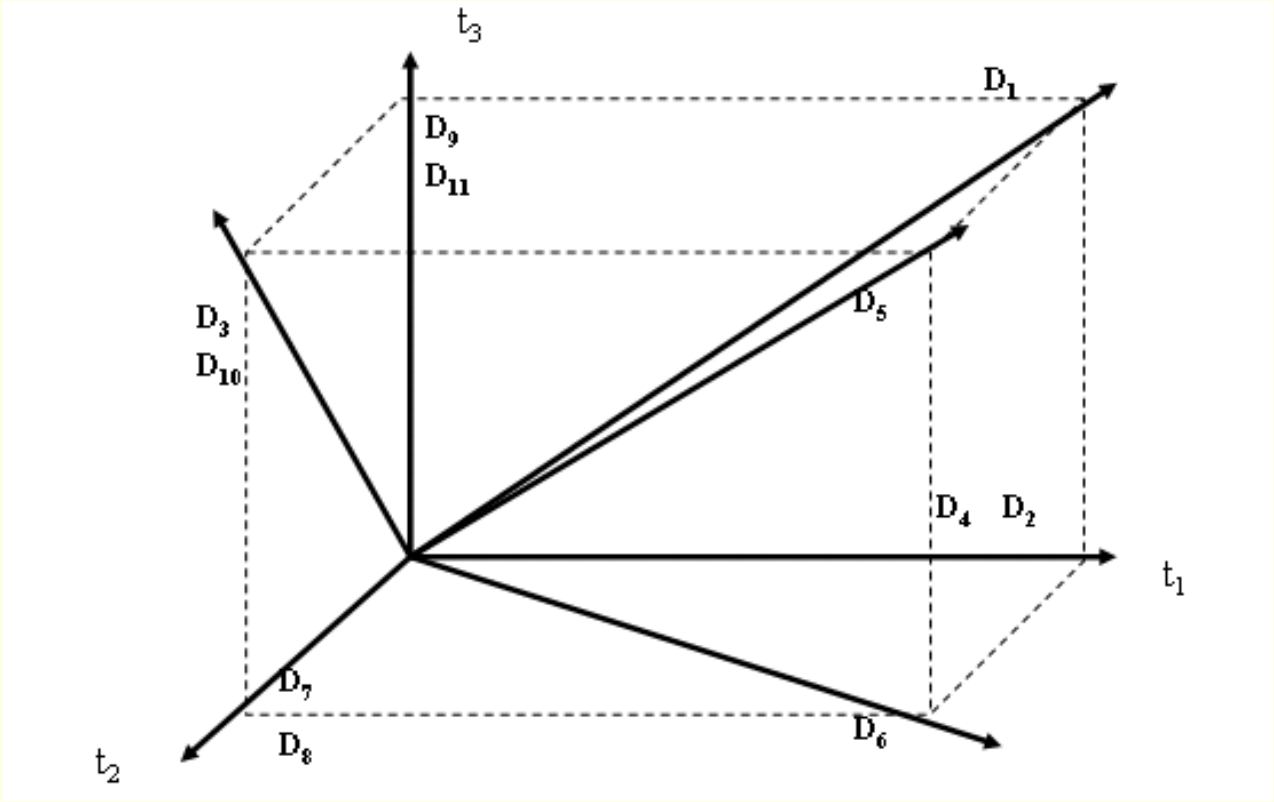
Q is a query – also represented as a vector



Boolean term combinations

โดย q1,q2,q3 คือ ข้อสอบถาม
t1,t2,t3 เป็น เทอมของคำสำคัญหรือดรรชนี
D1,D2,…D11 คือเอกสาร

นำความสำคัญของคำและการ Match กันของคำระหว่างข้อสอบถามและเอกสารมาสร้างเป็นเวกเตอร์ได้ดังนี้



Serial Search

- การค้นหาอนุกรม(Serial Search)
- นั้นค่อนข้างจะใช้เวลา
- ใช้เป็นส่วนหนึ่งของการค้นหาข้อมูลในระบบที่ใหญ่
- วิธีการนี้แสดงถึงการใช้ Matching Function

การค้นหาเอกสารทำได้โดย

- จะดึงเอกสารที่มีค่ามากกว่า Threshold ที่ถูกกำหนดมาให้อย่างเหมาะสมของ Matching Function นั้น แล้วทิ้งส่วนที่มีค่าน้อยกว่า ถ้า T เป็น Threshold ดังนั้น เซ็ทของเอกสารที่จะถูกดึงออกมาก็จะได้แก่

  Set{Di | M(Q,Di) >= T}

- เอกสารถูกจัดลำดับตามค่าที่เพิ่มขึ้นของ Matching Function เลือก Rank Position R ให้เป็น Cut-off และทุกๆ เอกสารภายใต้ Rank นี้จะถูกดึงออกมา เพื่อว่า

  B = { Di | r(i) <= R }

  ซึ่ง r(i) เป็น Rank Position ที่ถูกกำหนดให้ Di เป็นที่คาดหวังว่าเอกสารที่เกี่ยวข้องกับข้อเรียกร้องจะถูกบรรจุอยู่ในเซ็ทของเอกสารที่ถูกดึงออกมา (Retrieved Set)

ตัวอย่าง

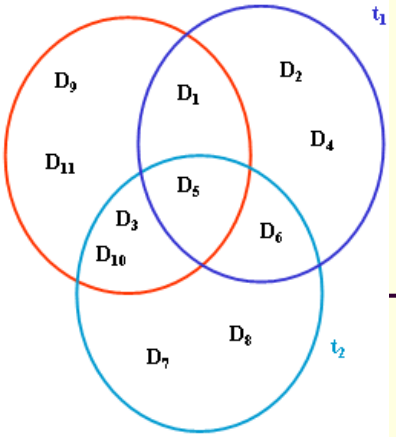
- ระบบหนึ่งมีเอกสาร Di อยู่ N จำนวน
- Serial Search จะทำได้โดยการคำนวณ N ค่าของ

M(Q,Di) สำหรับ I = 1 ถึง N

หรือกล่าวได้ว่า Matching Function นั้นจะถูกประเมินค่าที่แต่ละเอกสารสำหรับข้อคำถาม Q เดียวกัน

docs	t1	t2	t3	RSV=Q.Di
D1	1	0	1	4
D2	1	0	0	1
D3	0	1	1	5
D4	1	0	0	1
D5	1	1	1	6
D6	1	1	0	3
D7	0	1	0	2
D8	0	1	0	2
D9	0	0	1	3
D10	0	1	1	5
D11	0	0	1	4
Q	1	2	3	
	q1	q2	q3	

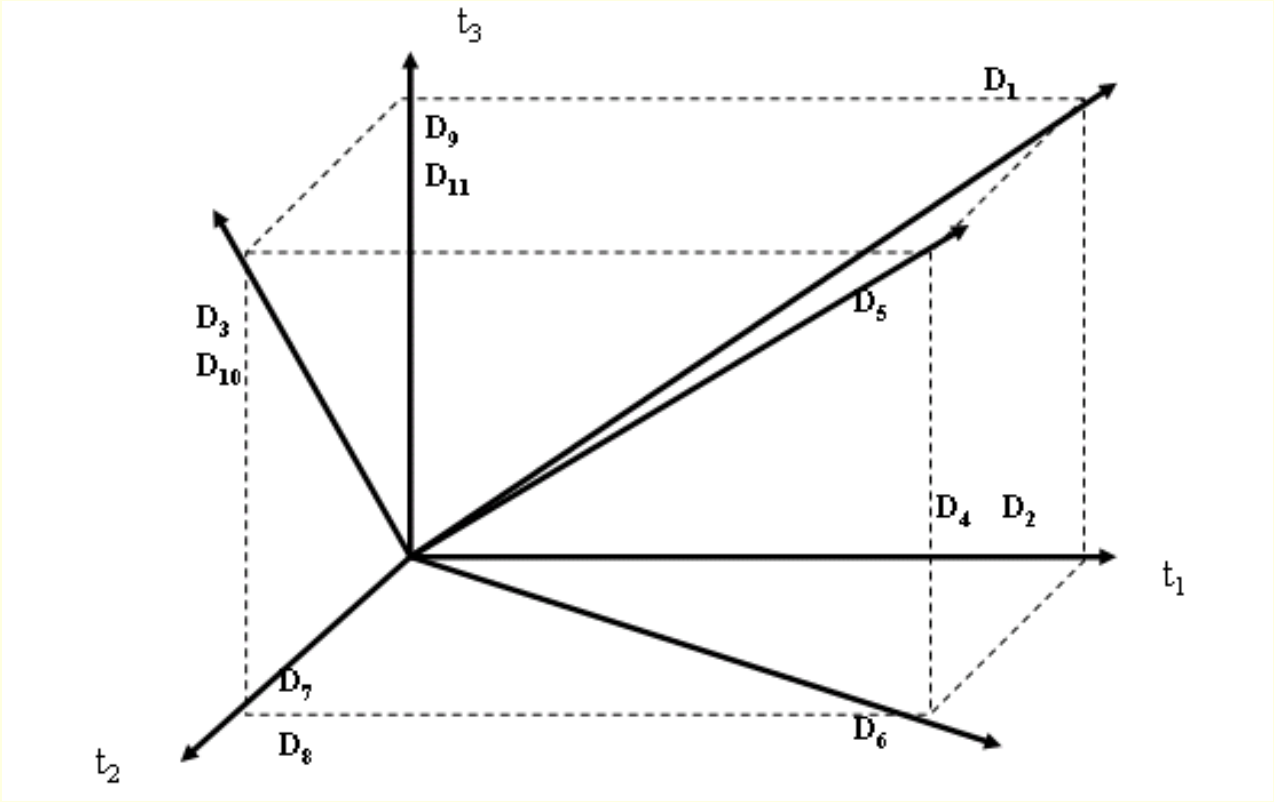
Q is a query – also represented as a vector



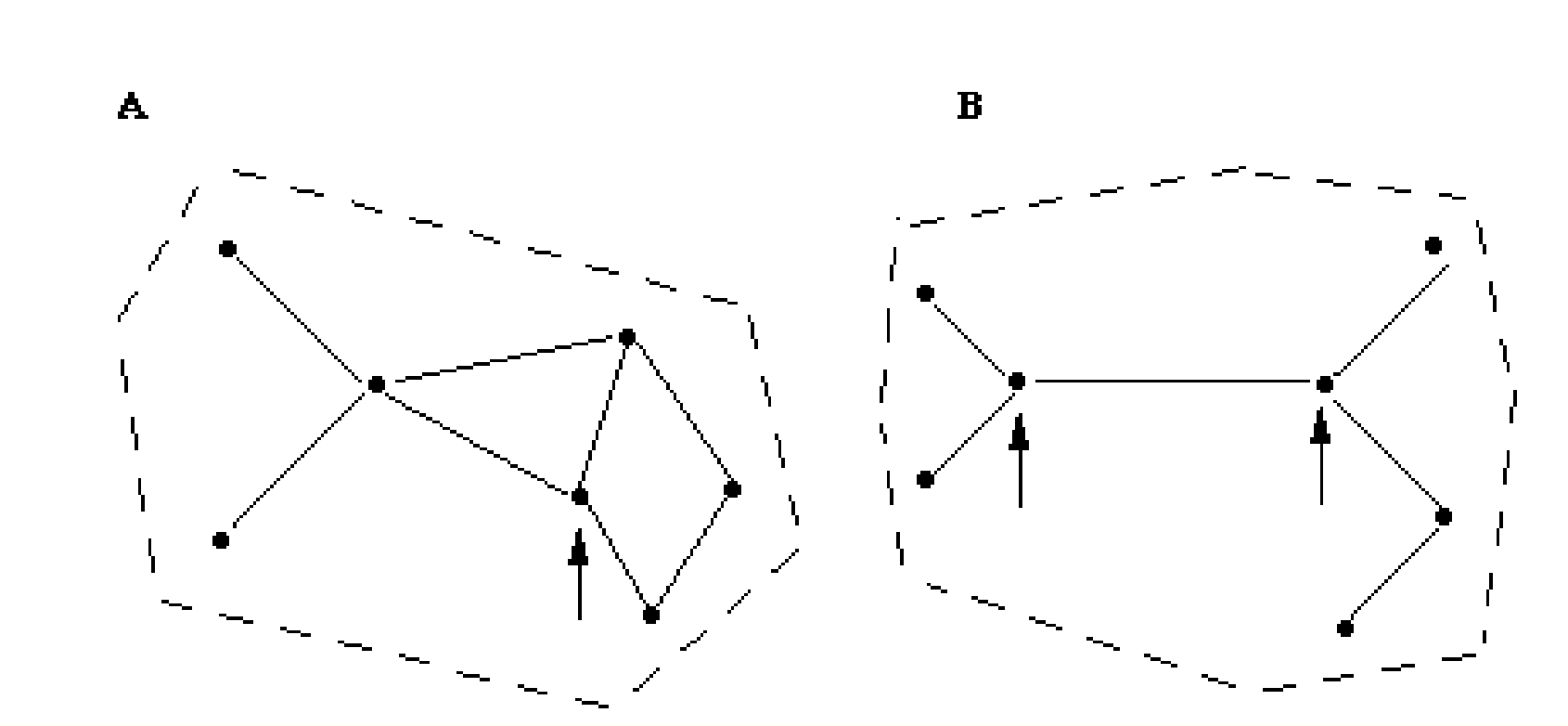
Boolean term combinations

โดย q1,q2,q3 คือ ข้อสอบถาม
t1,t2,t3 เป็น เทอมของคำสำคัญหรือดรรชนี
D1,D2,…D11 คือเอกสาร

นำความสำคัญของคำและการ Match กันของคำระหว่างข้อสอบถามและเอกสารมาสร้างเป็นเวกเตอร์ได้ดังนี้



ตัวแทนคลัสเตอร์ (Cluster Representatives)



แสดงตัวอย่างของ **Maximally Linked Documents**

ตัวแทนคลัสเตอร์ (Cluster Representatives)

- การหาตัวแทนที่เป็นการเฉลี่ย Descriptions ของสมาชิกภายในคลัสเตอร์เหล่านั้น ซึ่งจะทำได้โดยการคำนวณหาจุดศูนย์ถ่วง (Centroid) ของคลัสเตอร์หนึ่ง

ตัวแทนคลัสเตอร์ (Cluster Representatives)

- สมมุติว่า{ D1, D2, ... , Dn}เป็นเอกสารในคลัสเตอร์หนึ่ง

และแต่ละ Di นั้นถูกแสดงโดย Numerical Vector (d1, d2, ... , dt)
ซึ่งจุดศูนย์ถ่วงหรือ Centroid C ของคลัสเตอร์นั้น จะถูกกำหนดได้ดังนี้

$$C = \frac{1}{n} \sum_{i=1}^{n} \frac{D_i}{\| D_i \|}$$ (n เป็นจำนวนของเอกสารในคลัสเตอร์)

||Di|| นั้นโดยปกติก็เป็น Euclidean Norm, กล่าวคือ

$$\|Di\| = \sqrt{d12 + d22 + d32 + \ldots + dt2}$$

$\frac{1}{n}$

ตัวแทนคลัสเตอร์ (Cluster Representatives)

- แสดงแทนโดย Numerical Vector แต่จะแสดงแทนโดยไบนารีเว็คเตอร์ (Binary Vector) หรือคีย์เวิร์ดกลุ่มหนึ่ง

Di เป็น ไบนารี่เว็คเตอร์ที่มีค่าดังต่อไปนี้

Di = 1 ถ้ามีคีย์เวิร์ดอันที่ j ในเอกสารนี้

0 ถ้าไม่มีคีย์เวิร์ดอันที่ j ในเอกสารนี้

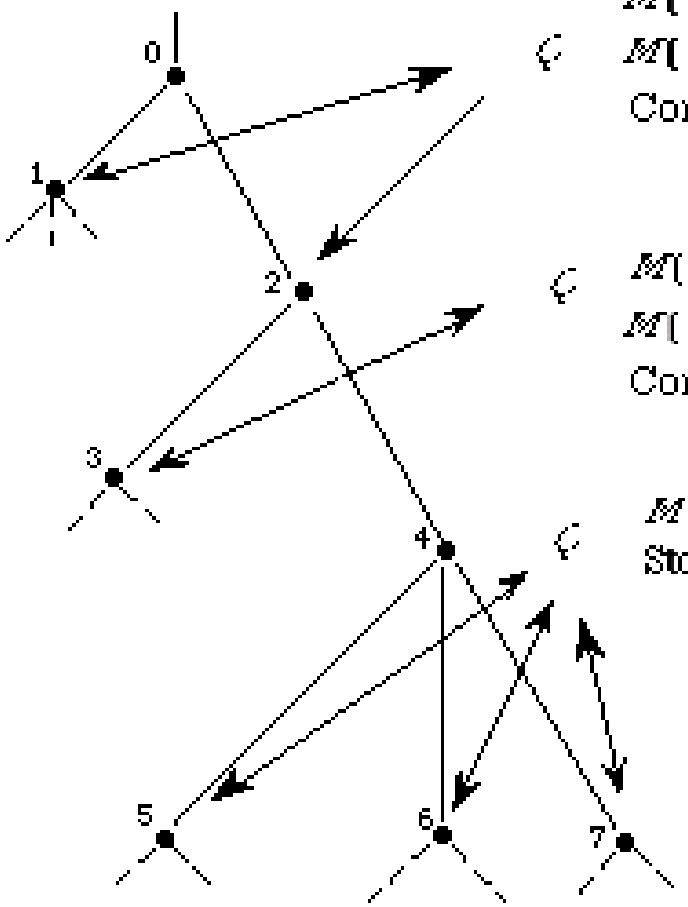
ดังนั้น $S = \sum_{i=1}^{n} D_i$ (n เป็นจำนวนของเอกสารในคลัสเตอร์นั้น)

การดึงข้อมูลโดยอาศัยคลัสเตอร์เป็นพื้นฐาน (Cluster-Based Retrieval)

- ต้องอาศัยข้อสมมุติฐานของคลัสเตอร์ (Cluster Hypothesis) ที่กล่าวว่า

“<u>เอกสารที่มีความใกล้ชิดคล้ายคลึงกัน ย่อมมีแนวโน้มที่จะเกี่ยวข้องกับข้อคำถามอันเดียวกัน</u> “

Cluster-Based Retrieval



$M(C,2) > M(C,1)$
$M(C,2) > M(C,0)$
Continue

$M(C,4) > M(C,3)$
$M(C,4) > M(C,2)$
Continue

$M(C,5), M(C,6), M(C,7) < M(C,4)$
Stop. Retrieve cluster 4

การที่อนุญาตให้มีการค้นหาลงไปได้มากกว่าหนึ่งกิ่งของทรี ซึ่งจะทำให้สามารถดึงข้อมูลที่ต้องการได้มากกว่าหนึ่งคลัสเตอร์ ด้วยความจำเป็น Decision Rule และ Stopping Rule

Interactive Search Formulation

ระบบควรเตรียมสิ่งต่อไปนี้ให้แก่ผู้ใช้ระบบในการสร้างข้อคำถามสำหรับค้นหาข้อมูล

1. ความถี่ของการเกิดขึ้นของค่าที่ใช้ในการค้นหาภายในฐานข้อมูลนั้น
2. จำนวนของเอกสารที่เป็นไปได้ที่จะถูกดึงออกมาโดยข้อคำถามของเขา
3. ค่าต่างๆ ที่เกี่ยวข้องกับค่าที่ใช้ในการค้นหาข้อมูล
4. ตัวอย่างของ Citations(ข้ออ้างอิง) ที่อาจจะถูกดึงออกมา
5. ค่าที่ถูกใช้ในการทำดัชนีของ Citations ใน 4. เป็นต้น

Feedback

- อธิบายถึงกลไกที่ระบบหนึ่งสามารถที่จะที่จะพัฒนาการทำงาน (Performance) ของระบบบนงานหนึ่ง โดยการนำเอาการทำงานที่ผ่านไปแล้วมาพิจารณา
- หรืออาจจะกล่าวได้ว่า ระบบอินพุท- เอาท์พุท (Input-Output) แบบง่ายๆ นี้จะให้ผลลัพธ์กลับมาเกี่ยวกับข้อมูลต่างๆ จากผลลัพธ์ที่ได้ เพื่อว่าจะสามารถที่จะถูกนำไปใช้ในการพัฒนาการทำงานบนอินพุทอันถัดไป

ระบบสืบค้นเนมเพจ
(Named Pages Finding System

- ระบบสืบค้นข้อมูลที่ผ่านมาในอดีตได้ใช้กลยุทธ์การค้นหาแบบตรงตามหัวข้อ (Subject Search Stratergy)
- ซึ่งมีหลักในการค้นหาให้พบเอกสารที่ไม่เกี่ยวข้องให้น้อยที่สุด
- ซึ่งกลยุทธ์ในการสืบค้นรูปแบบดังกล่าวจะ**ให้ผลการค้นหาที่มีจำนวนมากและยากต่อการเลือกคำตอบของผู้ใช้**

- กลยุทธ์การค้นหาแบบสิ่งที่เคยพบมาก่อนแล้ว(known-Item Search Strategy)
- กลยุทธ์นี้มีหลักการใน**การค้นหาโดยใช้คำสำคัญ**จากผู้ใช้ที่สามารถระบุได้ถึงความหมาย ชื่อ หรือเนื้อหา โดยสรุปของเอกสารที่ต้องการ
- และค้นหาเอกสารที่ตรงกับคำสำคัญมากที่สุดเพียง **1-2 เอกสารเป็นคำตอบให้กับผู้ใช้** กลยุทธ์การค้นหานี้จะช่วยให้ผู้ใช้สามารถค้นหาคำตอบจากการสืบค้นได้อย่างง่ายดาย

ระบบสืบค้นเนมเพจ
(Named Pages Finding System

- ระบบเริ่มต้นทำงานในขั้นตอนการเตรียมข้อมูล(Data Preparation Phase) จากส่วนสกัดข้อมูล (Data Extraction Module) ส่วนสกัดข้อมูลทำหน้าที่คัดแยกส่วนของข้อมูลที่ต้องการออกจากเอกสาร เช่นคัดแยกเฉพาะส่วนหัวข้อหรือส่วนคำบรรยายลิงค์เป็นต้น ข้อมูลที่คัดแยกเรียบร้อยแล้วถูกเรียกว่า แหล่งตัวแทนเอกสาร(Source of Information) ซึ่งเป็นข้อมูลที่พร้อมนำไปวิเคราะห์ในส่วนการวิเคราะห์ชื่อที่ปรากฏเด่นชัด (Frequent Name Analysis Module)
- ส่วนการวิเคราะห์ชื่อที่ปรากฏเด่นชัดทำหน้าที่วิเคราะห์หาชื่อที่ปรากฏบ่อยครั้งในแหล่งตัวแทนเอกสาร ชื่อที่วิเคราะห์ได้จะถูกจัดเก็บเป็นตัวแทนชื่อเอกสารที่ปรากฏเด่นชัด (Frequent Name Representator) เพื่อใช้สำหรับการสืบค้นเนมเพจในลำดับถัดไป

ระบบสืบค้นเนมเพจ (Named Pages Finding System

- ในขั้นตอนการสืบค้นเนมเพจ(Named Page Retrievel Phase) เริ่มต้นจากระบบรับคำถาม(Query) จากผู้ใช้งานและจัดรูปแบบให้อยู่ในรูปแบบเดียวกันกับตัวแทนชื่อเอกสาร โดยส่วนการเตรียมคำถาม(Query Formulation Module)
- คำถามที่ได้จะถูกนำไปเปรียบเทียบความใกล้เคียงกับเอกสารที่ปรากฏเด่นชัด โดยส่วนสืบค้นเนมเพจ(Named Page Retrieval Module)
- ส่วนสืบค้นเนมเพจจะให้ผลลัพธ์การสืบค้นเป็นเอกสารที่มีตัวแทนเอกสารใกล้เคียงกับคำถามของผู้ใช้มากที่สุด

Boolean queries: Exact match

- The Boolean retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean Queries use *AND, OR* and *NOT* to join query terms
    - Views each document as a set of words
    - Is precise: document matches condition or not.
  - Perhaps the simplest model to build an IR system on
- Primary commercial retrieval tool for 3 decades.
- Many search systems you still use are Boolean:
  - Email, library catalog, Mac OS X Spotlight

Example: WestLaw http://www.westlaw.com/

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- Tens of terabytes of data; 700,000 users
- Majority of users *still* use boolean queries
- Example query:
  - What is the statute of limitations in cases involving the federal tort claims act?
  - LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM
    - /3 = within 3 words, /S = in same sentence

Example: WestLaw http://www.westlaw.com/

- Another example query:
  - Requirements for disabled people to be able to access a workplace
  - disabl! /p access! /s work-site work-place (employment /3 place)
- Note that SPACE is disjunction, not conjunction!
- Long, precise queries; proximity operators; incrementally developed; not like web search
- Many professional searchers still like Boolean search
  - You know exactly what you are getting
- But that doesn't mean it actually works better….

Boolean queries: More general merges

- Exercise: Adapt the merge for the queries:

  Brutus *AND NOT* ***Caesar***

  Brutus *OR NOT* ***Caesar***

Can we still run through the merge in time O(x+y)?

What can we achieve?

Merging

What about an arbitrary Boolean formula?

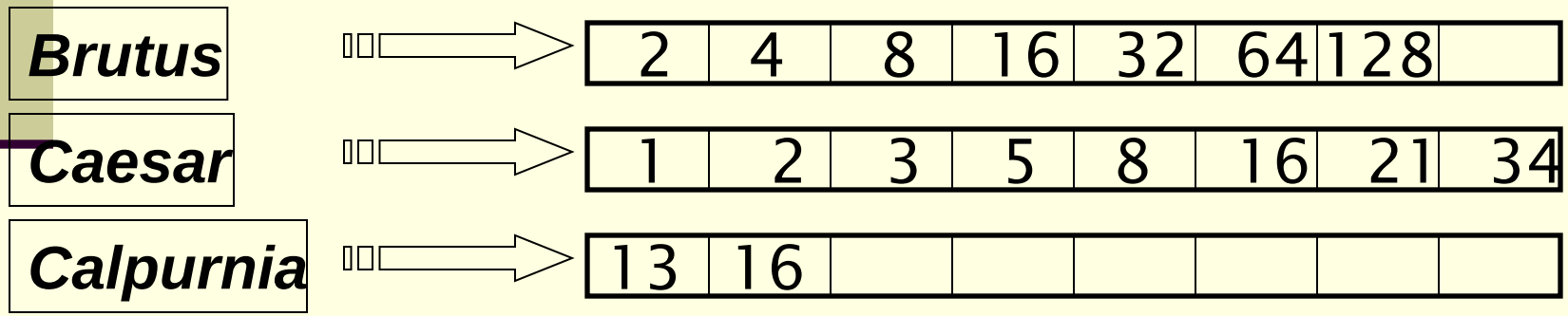
(Brutus *OR* ***Caesar)*** *AND NOT*

(Antony *OR* ***Cleopatra)***

- Can we always merge in “linear” time?
  - Linear in what?
- Can we do better?

Query optimization

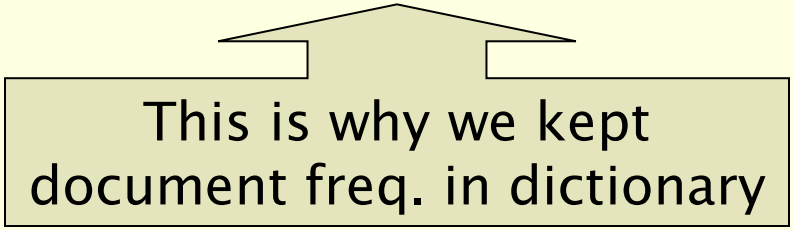
- What is the best order for query processing?
- Consider a query that is an *AND* of *n* terms.
- For each of the *n* terms, get its postings, then *AND* them together.

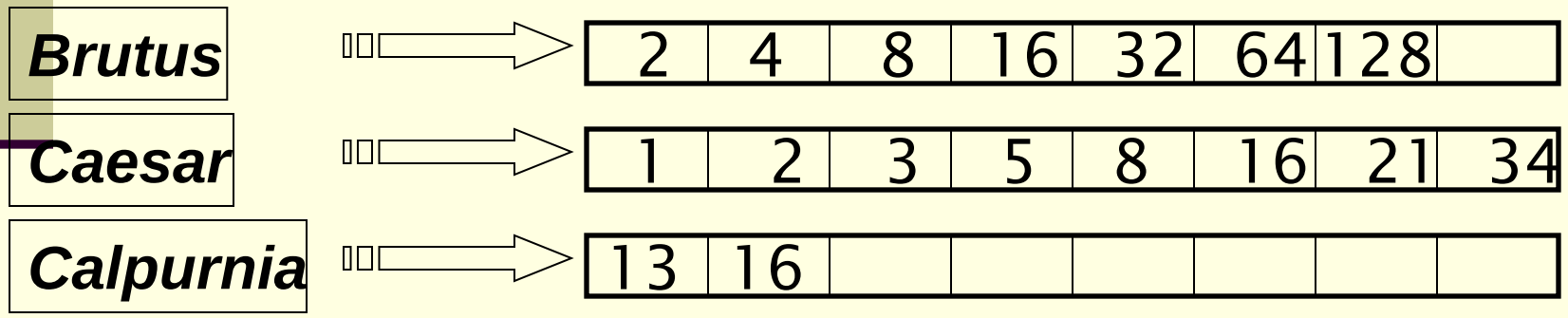


Query: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

Query optimization example

- Process in order of increasing freq:
  - *start with smallest set, then keep cutting further*.





Execute the query as (***Calpurnia*** *AND* ***Brutus)*** *AND* ***Caesar***.

More general optimization

- e.g., *(**madding** OR **crowd**) AND (**ignoble** OR **strife**)*
- Get doc. freq.'s for all terms.
- Estimate the size of each *OR* by the sum of its doc. freq.'s (conservative).
- Process in increasing order of *OR* sizes.

Exercise

- Recommend a query processing order for

***(tangerine** OR **trees)** AND*
***(marmalade** OR **skies)** AND*
(kaleidoscope** OR **eyes)

Term	Freq
eyes	213312
kaleidoscope	87009
marmalade	107913
skies	271658
tangerine	46653
trees	316812

Query processing exercises

- Exercise: If the query is ***friends*** *AND* ***romans*** *AND (NOT* ***countrymen****)*, how could we use the freq of ***countrymen***?
- Exercise: Extend the merge to an arbitrary Boolean query. Can we always guarantee execution in time linear in the total postings size?
- Hint: Begin with the case of a Boolean *formula* query where each term appears only once in the query.

Exercise

- Try the search feature at http://www.rhymezone.com/shakespeare/
- Write down five search features you think it could do better