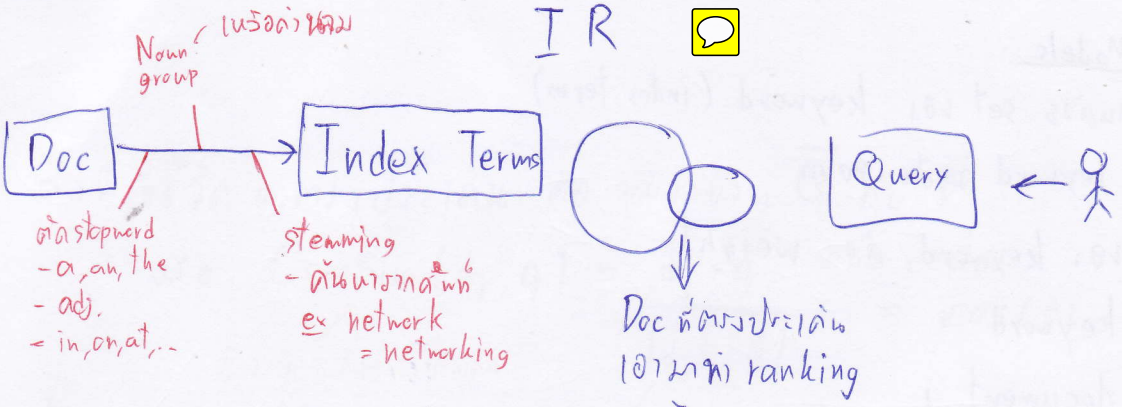
IR



- ระบบ IR → จัดเก็บอย่างไร, เรียกค้นอย่างไร, การจัดอันดับ
  → relevance (ความสัมพันธ์) ที่ดีที่สุด
  → ใช้ Keyword (Index term) ในการ query
- พวก keyword จะถูกเก็บใน inverted file บอกว่า keyword นี้อยู่ใน Doc ไหนบ้าง
- Ranking คือการเรียงอันดับ Doc ตาม ค. ความสัมพันธ์
- User Task
  - Retrival : ใช้ query ที่หา doc ที่ต้องการ
  - Browsing : ไม่มีจุดหมาย คลิก link ไปเรื่อยๆ
    - Model
      - Flat
      - Structure Guided
      - Hypertext
    - Adhoc : Doc มีจำนวนคงที่ แต่ query เปลี่ยน
    - Filtering : query คงที่ แต่มี doc ใหม่ๆ เข้ามาเรื่อยๆ

IR Model ที่เรียน

- Classic Model (ให้น้ำหนักของ ค. ที่เจอคำนั้นเท่ากัน)
  - boolean
  - vector
  - probabilistic
  - fuzzy
- Structured Model (ให้น้ำหนักไม่เท่ากัน เช่น หน้าปก, หัวเรื่อง, เนื้อเรื่อง)
  - Non-Overlapping Lists
  - Proximal Nodes

• Classic IR Models

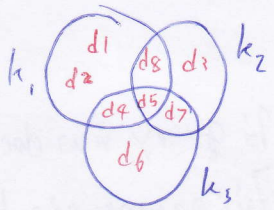
- Doc แทนด้วย set ของ keyword (index term)
- ส่วนใหญ่ keyword จะเป็น noun
- ค่าความสำคัญของ keyword คือ weight
- k_i = keyword i
- d_j = document j
- w_{ij} = weight ของ keyword i ใน document j

Boolean Model

- ค่า weight เป็น 0 หรือ 1
- เอา keyword มาเชื่อมด้วย AND($\wedge$) OR($\vee$) NOT($\neg$) เป็น Query

$$q = k_a \wedge (k_b \vee \neg k_c)$$ *เรียกว่า boolean expression

- การค้นหาเอกสาร ทำโดยการเขียนแผนภาพ



- ไม่มีการทำ Ranking
- มักจะได้ doc ตอบที่มากเกิน หรือน้อยเกินไป

Vector Model

- ใช้ w ของแต่ละ keyword ในเอกสาร
- มี w ใน query
- ตัว doc หรือ query จะเป็นตาราง หรือ vector เช่น

	k_1	k_2	k_3
d1	2	0	1
d2	1	0	0
q	1	2	3

$\Rightarrow$

$\vec{d_1} = (2, 0, 1)$
$\vec{d_2} = (1, 0, 0)$
$\vec{q} = (1, 2, 3)$

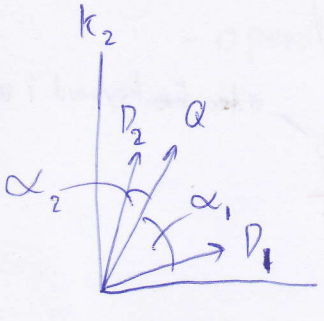
- สิ่งที่ใช้วัด ค.เหมือนประเด็น คือ มุมระหว่าง $\vec{Q}$ กับ $\vec{D}$

หรือ $Cos\,Sim(d_j, q) = \dfrac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \cdot |\vec{q}|} = \cos(\theta)$

ผลคูณ vector

ยิ่งมาก... ยิ่งเหมือนประเด็น

ขนาด vector



$D_1 = (0.8, 0.3)$
$D_2 = (0.2, 0.7)$
$Q = (0.4, 0.8)$

$$sim(Q, D_1) = \frac{0.8(0.4) + 0.3(0.5)}{\left(\sqrt{0.8^2+0.3^2}\right)\left(\sqrt{0.4^2+0.8^2}\right)}$$

$$= 0.74$$

- การคำนวน weight ของ keyword ในแต่ละ doc / query

- ใช้ tf = term frequency ใน doc ๆ doc

idf = inverse document freq. เพื่อลดความสำคัญของคำที่ปรากฏในหลายๆ doc ในระบบ

สูตร $W_{ij} = tf(i,j) \times idf(i)$

W_{ij}: keyword, doc*

$$tf(i,j) = \frac{freq(i,j)}{\max(freq(i,j))}$$

freq(i,j): จำนวนครั้งที่ keyword i ปรากฏในเอกสาร j

$$idf(i) = \log\left(\frac{N}{n_i}\right)$$

N: # doc ในระบบ

n_i: # doc ที่มี keyword i อยู่

- คำนวน weight ของ query

$$W_{i,q} = 0.5 + \frac{0.5 \times freq(i,q)}{\max(freq(i,q))} \times \log\left(\frac{N}{n_i}\right)$$

ดูจากเอกสารในระบบ

ex Query = "Visitor at your door or my door"

key terms = {"visitor", "door", "door"}

freq(visitor, q) = 1 , freq(door, q) = 2

$$W_{door,q} = 0.5 + \frac{0.5 \times 2}{2} \times \log\left(\frac{6}{2}\right)$$

- **Probabilistic Model**
  - ให้ค่าน่าจะเป็นที่ผู้ใช้จะเจอ doc ที่ตรงประเด็น
  - ประสิทธิภาพจะเพิ่มขึ้นเรื่อยๆ โดยคนต้องบอกว่า doc ที่ออกมาตรง/ไม่ตรง
  - รูปแบบ Fuhr Model

$$Sim(d_j, q) = \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1-P(k_i|R)} + \log \frac{1-P(k_i|\bar{R})}{P(k_i|\bar{R})} \right)$$

โดย $P(k_i|R) = \frac{V_i + \frac{n_i}{N}}{V+1}$

- V_i = # doc ที่ตรงประเด็นและมี keyword i อยู่
- n_i = # doc ในระบบ
- V = # doc ที่ make ว่าตรงประเด็น
- $P(k_i|R)$ = prob. ที่ keyword i จะอยู่ในเอกสารที่ตรงประเด็น

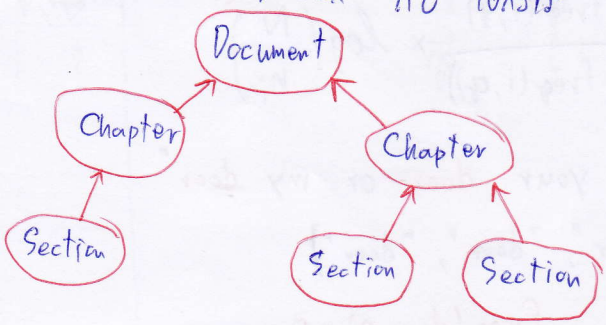
$$P(k_i|\bar{R}) = \frac{n_i - V_i + \frac{n_i}{N}}{N - V + 1}$$

- $P(k_i|\bar{R})$ = prob ที่ k_i อยู่ใน doc ที่ไม่ตรงประเด็น
- n_i = # doc ที่มี keyword i อยู่

- **Structured Text Model**
  - word ที่อยู่ใน title มี weight มากกว่า word ที่อยู่ในเนื้อเรื่อง
  - น้ำหนักของคำขึ้นอยู่กับตำแหน่งของคำๆนั้น
  - มี 2 วิธี

1) **Non-Overlapping Lists**
   - สร้าง List ของ ← Chapter, Section, SubSection, ... เก็บค่า text region ว่าอยู่ตรงแนวที่เท่าไหร่ ถึงเท่าไหร่
   - ค้นหาที่ query ได้มีข้อจำกัด

2) **Proximal Nodes**
   - ใช้ hierarchical index กับเอกสาร



• Fuzzy Logic

- Fuzzy Set = set ของเหตุการณ์ที่ไม่แน่นอน
- ค่าความเป็นสมาชิก มีค่าระหว่าง 0 - 1
- operator
  - complement $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$
  - intersection $\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x)$
    $= \min(\mu_A(x), \mu_B(x))$
  - union $\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x)$
    $= \max(\mu_A(x), \mu_B(x))$
- ค่าความสัมพันธ์ระหว่าง index term (keynord)

$$C_{i,j} = \frac{n_{i,j}}{n_i + n_j - n_{i,j}}$$

$n_{i,j}$ = # doc ที่มี i และ j

$C_{i,j}$ = ค่าความสัมพันธ์ i กับ j

n_i = # doc ที่มี keynord i

- ต้องแปลงประโยค query เช่น "I would like cat and have dog" ให้เป็น index term เช่น "cat", "dog"
- จากนั้นหาค่าความสัมพันธ์ของ cat, dog กับ index term อื่นในระบบ ($C_{i,j}$)
- ex

Index 1	Index 2	Relate
Dog	Cat	0.33
~	Bird	0.5
~	Zebra	~
~	Zoo	~
~	Puppy	~
~	House	~
~	Robber	~

- หาระดับ ค.เป็นสมาชิก ของ index term ในเอกสารใดๆ

ex $dog = \{ d_1 \to 1, d_2 \to 0.33, d_3 \to 1, d_4 \to 0 \}$

d_1 มี dog ใน d_1

d_2 ไม่มี dog แต่ไปดูในตาราง ค.ร่วมพ้นธ์ เจอ cat มีค่า $c = 0.33$ เวลาเอามา (ถ้ามี c หมายถึงให้เอาค่าของคุณมาใช้)

$d_1 = \{ dog, cat, bird, zebra, zoo \}$

$d_2 = \{ cat, kitty, fish \}$

$d_3 = \{ dog, puppy, house, robber \}$

$d_4 = \{ ant, sugar \}$

- จากนั้นทำ operator ตามที่โจทย์ถาม

- union → $\mu_{dog \cup cat}(x) = (1 - (1 - \mu_{dog}(x)) * (1 - \mu_{cat}(x)))$

- intersect → $\mu_{dog \cap cat}(x) = \mu_{dog}(x) \cdot \mu_{cat}(x)$

- complement → $\mu_{\overline{dog} \cap \overline{cat}}(x) = (1 - \mu_{dog}(x)) \cdot (1 - \mu_{cat}(x))$

- แทน x ด้วย d_1, d_2, d_3, d_4 จะได้ค่าความเป็นสมาชิก (ค.ตรงประเด็น) แล้วมา ranking จากมากไปน้อย

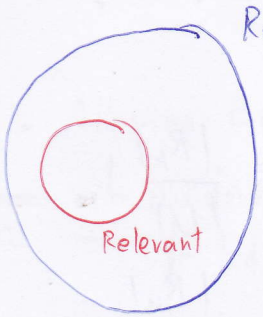
• Evaluation of IR system

- วัดความตรงประเด็นของ doc ที่ออกมาให้ค่า Recall กับ Precision

- $\text{Recall} = \frac{\text{\# doc ที่ออกมาแล้วตรงประเด็น}}{\text{\# doc ที่ตรงประเด็นทั้งหมด}}$ = ค.สามารถในการดึง doc ที่ตรงประเด็นทั้งหมดออกมา

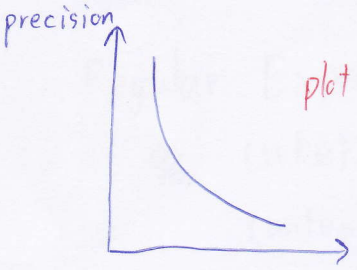
- $\text{Precision} = \frac{\text{\# doc ที่ออกมาแล้วตรงประเด็น}}{\text{\# doc ที่ดึงออกมา}}$ = ค.สามารถในการดึง เฉพาะ doc ที่ตรงประเด็นออกมา

- การวัด recall ทำยาก เพราะหาจำนวน doc ที่ตรงประเด็นทั้งหมดในระบบไม่ได้



recall ↑ , precision ↓

- Precision / Recall Curves



plot โดยวัดค่า precision ที่ recall ต่างๆ

- ดูวิธีวัด recall, precision ใน slide
- ค่าที่ใช้วัดประสิทธิภาพ
  - Average Precision : ค่าเฉลี่ยของ precision เฉพาะตอนที่ recall เพิ่ม
  - R-Precision : ค่า precision ที่ตำแหน่ง R โดย R = # doc ที่ตรงประเด็น
    - Precision Histogram → การเปรียบเทียบ algorithm A กับ B

      โดย $RP_{A/B}(i) = RP_A(i) - RP_B(i)$
  - F-Measure : ใช้ harmonic mean ระหว่างค่า recall และ precision

    $$F = \frac{2}{\frac{1}{R} + \frac{1}{P}}$$

    * หา F เฉพาะตอนที่ recall เปลี่ยนแปลง แล้วเอามาเฉลี่ย
  - E-Measure : คือ F-Measure ที่มีการใช้ weight ระหว่าง precision / recall

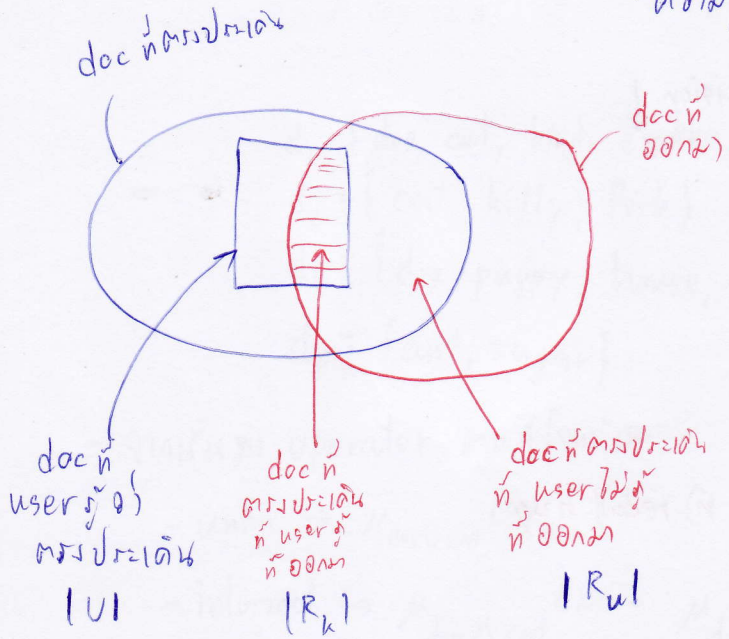
    $$E = \frac{(1 + \beta^2)}{\frac{\beta^2}{R} + \frac{1}{P}}$$

    โดย $\beta = 1$ → weight เท่ากัน

    $\beta > 1$ → ~ precision มากกว่า

    $\beta < 1$ → ~ recall มากกว่า

- User-Oriented Measure
  - ดูว่าสิ่งที่ user รู้เป็นอย่างไร
    - ครอบคลุมสิ่งที่ user รู้อยู่แล้ว
    - ความรู้ใหม่ (doc ที่ตรงประเด็นแต่ user ไม่รู้)



$$\text{coverage} = \frac{|R_k|}{|U|}$$

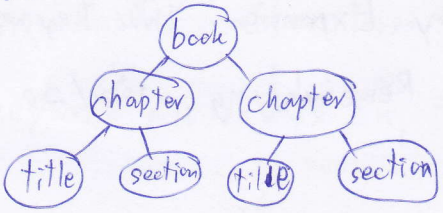
$$\text{Novelty} = \frac{|R_u|}{|R_u| + |R_k|}$$

Query Languages

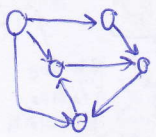
- Keyword-Base Querying
  1) Single-word query เช่น Apple
  2) Boolean-query : เอา single word มาเชื่อมกันด้วย Boolean operator
     - OR, AND, BUT เช่น Apple AND Banana
  3) Natural Language query : พิมพ์ประโยคเป็นคำถามปกติ
  4) Context query : มี word หลาย word
     - Phrasal query → หลายคำนั้นต่อกันเลย ex "information theory"
       - → อนุญาตให้มี stop words คั่น หรือ stemming ได้
       - → ต้องมี inverted index ที่เก็บตำแหน่งของ keywords ในเอกสาร
     - Proximity query → หลายคำนั้นห่างกันได้
       - → ต้องกำหนดระยะห่างสูงสุดระหว่างคำ

- Pattern Matching
  - match กับคำอื่นที่ไม่เหมือนกัน 100% ได้
  - Simple Pattern ← Prefix, Suffix, Substring, Range
  - Allowing Errors เช่น การระบุผิด
    - การวัดความต่างกันของ word มี
      - Edit distance (Levenstein distance)
        - จำนวนครั้งที่ต้องลบ, เพิ่ม, แทนที่ เพื่อให้ string เหมือนกัน
      - Longest Common Subsequence
        - ความยาวสูงสุดของตัวอักษรที่เรียงกัน
  - Regular Expression
    - ex (ule) nable(eling)
      (unlen)* able

- Structural Query
  - ใน query มีทั้งตัว content และโครงสร้าง
    - content: word, phrase, pattern
    - โครงสร้าง: หาใน field ใน
  - โครงสร้างมี 3 ประเภท
    1) Fixed structure : มี set of field ที่คงที่ เช่น หนังสือ, email



    2) Hypertext ; เป็น directed graph ใน node มี text อยู่



    3) Hierarchical Structure

Query Operation

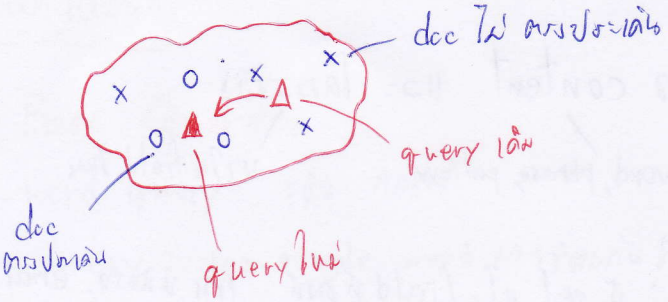
- ปัญหาของตัว query < วัน/เก่นไป
  กำกวม

- วิธีการปรับปรุงตัว query
  - user feedback
    - Rocchio Method
    - Ide Regular Method
    - Relevance Feedback
  - automatically
    - Global Analysis – ใช้ doc ทั้งหมดในระบบ
    - Local Analysis – ใช้ doc ที่ถูก return มาลัลอย่างเดียว
      - Association Cluster
      - Metric Cluster
      - Scalar Cluster

เปลี่ยน weight เพิ่ม keyword

Relevance Feedback

- เมื่อ query แล้ว user ต้อง mark ว่า เอกสารไหน ตรง / ไม่ตรง ประเด็น
- ปรับ query ให้เข้าใกล้ doc ที่ตรงประเด็น



- การเปลี่ยนรูป query
  - Query Expansion: เพิ่ม keyword ใหม่
  - Term Reweighting: เพิ่ม/ลด w

* วิธี

1) Rocchio Method

$$Q_1 = \alpha Q_0 + \frac{\beta}{n_1}\sum_{\forall d_j \in D_r} \vec{d}_j - \frac{\gamma}{n_2}\sum_{\forall d_j \in D_n} \vec{d}_j$$

- n_1 จำนวน doc ที่ตรงประเด็น
- $\sum_{\forall d_j \in D_r} \vec{d}_j$ Sum $\vec{d}$ ที่ตรงประเด็น
- n_2 จำนวน doc ที่ไม่ตรงประเด็น
- $\sum_{\forall d_j \in D_n} \vec{d}_j$ Sum $\vec{d}$ ที่ไม่ตรงประเด็น

α = การให้น้ำหนักกับ query ดั้งเดิม
β = ~ doc ที่ตรงประเด็น
γ = ~ doc ที่ไม่ตรงประเด็น

2) Ide Regular Method

$$\vec{q}_1 = \alpha \vec{q}_0 + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

- เหมือนอันที่แล้ว แต่ไม่มี n_1, n_2 มาหาร

3) Relevance Feedback

$$\vec{q}_m = \vec{q} + \alpha \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \beta \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

- เหมือนอันที่แล้ว แต่ไม่มี ตัวคูณหน้า $\vec{q}_0$

* ข้อเสียของวิธี user feedback (Relevance Feedback) คือ user mark ได้ว่า ตรง/ไม่ตรง ประเด็น เฉพาะ เอกสาร ที่ return ออกมาเท่านั้น ทำให้ query ตัวใหม่ ไม่ได้เข้าหา doc ที่ตรงประเด็น แต่ไม่ถูก return ออกมาด้วย

- Create new query automatically

Global Analysis

- ให้ doc ทั้งหมดในระบบ เอามาทำ Similarity Thesaurus
- คือ หา ความสัมพันธ์ระหว่าง term (keyword)
- แทน keyword ด้วย เวกเตอร์ $\vec{k}_i = (w_{i1}, w_{i2}, w_{i3}, w_{iN})$

w_{ij} ของ keyword i ใน doc j

- วิธีการ

$$w_{i,j} = \frac{\left(0.5 + 0.5 \frac{tf_{i,j}}{\max_k \{tf_{i,j}\}}\right) itf_j}{\sqrt{\sum_{k=1}^{N} \left[0.5 + 0.5 \frac{tf_{i,k}}{\max\{tf_{i,k}\}}\right]^2 itf_k^2}}$$

ความถี่ของ term k_i ใน doc j

ผลรวม term i ในทุกเอกสาร

$$itf_j = \log \frac{t}{t_j}$$

t = จำนวน term ทั้งหมด

t_j = จำนวน distinct keyword ใน doc j

- ความสัมพันธ์ระหว่าง term $C_{u,v} = \vec{k}_u \cdot \vec{k}_v$ *

  * $C_{u,v} = C_{v,u}$

- ค่าความเหมือน

$$sim(q, k_v) = \vec{q} \cdot \vec{k}_v = \sum_{k_u \in q} w_{u,q} \times C_{u,v}$$

- หลังจากเพิ่ม keyword ตามวิธีคิด sim ที่คิดออกมา ต้องนำ w ใหม่ ไปรวม

$$w_{v,q'} = \frac{sim(q, k_v)}{\sum_{k_u \in q} w_{u,q}}$$

(ผลรวม w.ป.ร. ใน q อันใหม่)

ex ค่า sim กับการเพิ่ม keyword

$sim(q, k_1) = 0.53$
$sim(q, k_2) = 0.36$
$sim(q, k_3) = 3.98$
$sim(q, k_4) = 1.87$

query เดิม $q = k_1 + k_4$

0.53 1.87

เงื่อนไข: add keyword ที่ค่า sim > 0.53

query ใหม่ $q' = k_1 + k_3 + k_4$

แล้วทำการคิด w ใน query ใหม่

Automatic Local Analysis : ใช้เฉพาะ doc ที่ return ออกมา
- เพิ่ม keyword ที่เกี่ยวข้องกับ keyword ที่ query เดิม
- การสร้าง local clusters
  1) Association cluster : ใช้ความถี่ของ keyword ในแต่ละเอกสาร
  2) Metric cluster : ใช้ระยะห่างระหว่าง keyword ~
  3) Scalar cluster : ความหมาย ใกล้เคียงกันเท่าไหร่

• Association Cluster
- แสดงเป็น matrix ของความสัมพันธ์ระหว่าง keyword หรือ Stem (รากศัพท์)
- เริ่มจากเขียน matrix ของความถี่ของ keyword ในแต่ละ Doc ที่ return ออกมา
- ทำ correlation matrix (c) โดย

$$C_{u,v} = \sum_{d_j \in D_l} f_{s_{u,j}} \times f_{s_{v,j}}$$

- แล้วทำ normalized correlation matrix (S)

$$S_{u,v} = \frac{C_{u,v}}{C_{u,u} + C_{v,v} - C_{u,v}}$$

เช่น

	A	B	C	D
A	1	0.7	0.18	0.44
B	0.7	1	0.85	0.63
C	0.18	0.85	1	0.63
D	0.44	0.63	0.63	1

จะได้ term relation
1. {A, B}
2. {B, C}
3. {C, B}
4. {D, B, C}

=> 1. {A, B}
2. {B, C}
3. {D, B, C}

เอาตัวมากสุดมาเขียน

query เดิม A ไม่ต้องแตกเป็น B เพราะ query เดิม มี B อยู่แล้ว

$q = A + B$ — B แตก C ออกไปได้ โดยค่า weight ของ C ก็คือ $S_{B,C}$

query ใหม่

$q = A + (B + 0.85\,C)$

- Metric Cluster
  - เอาระยะห่างระหว่าง keyword มาหาความสัมพันธ์ เป็น correlation matrix
  - แล้ว normalize

เช่น

	S_1	S_2	S_3
S_1	0	0.51	0.50
S_2	0.51	0	0.60
S_3	0.50	0.60	0

Stem Relation

1. $\{S_1, S_2\}$
2. $\{S_2, S_3\}$
3. $\{S_3, S_2\}$

$\Rightarrow$

1. $\{S_1, S_2\}$
2. $\{S_2, S_3\}$

ให้ $S_1 = \{A, B, C\}$, $S_2 = \{D, E\}$, $S_3 = \{F\}$

query ดั้งเดิม

$q = A + 2D$

A อยู่ใน S_1 / S_1 แทนไป S_2 ได้

D อยู่ใน S_2 / S_2 แทนไป S_3 ได้

query ใหม่

$q' = (S_1 + 0.51 S_2) + 2(S_2 + 0.6 S_3)$

$= S_1 + 2.51 S_2 + 1.2 S_3$

- วิธีหา

$$S_{u,v} = \frac{C_{u,v}}{|V(S_u)| \times |V(S_v)|}$$

ขนาดของความสัมพันธ์ที่ u

- Scalar Cluster
  - ให้ความหมายของ stem ที่ใกล้เคียงกัน เช่น Tree, Water, Fertillizer
  - ความเกี่ยวข้องกับคำอื่นให้ $\vec{S}_u = (C_{u,1}, C_{u,2}, \dots, C_{u,t})$

ค.สัมพันธ์ของ stem u กับ keyword ที่ 1

  - ต้องสร้าง scalar association matrix ค.สัมพันธ์ ระหว่าง stem

$$S_{u,v} = \frac{\vec{S}_u \cdot \vec{S}_v}{|\vec{S}_u| \times |\vec{S}_v|}$$

• Text Operation

- จาก full text ⟶ index term (keyword)
- Lexical Analysis : เปลี่ยน input character stream เป็น stream word or token (แบ่งเป็นคำๆ)
  เช่นการตัด hyphen
  state-of-the-art ⇒ state of the art

- การกำจัด stopword
  - ตัดพวก article, preposition, conjunction,... เพื่อลดขนาดของ indexing structure

- การทำ Stemming หารากศัพท์
  - คำในภาษาอังกฤษ เปลี่ยนรูปได้
  - มีวิธีการแบบอัตโนมัติ 4 วิธี
  1) Affix removal : ตัดคำที่เติมออก -ing, un-, -s
  2) Successor Variety : คำที่เป็นรากศัพท์ ดูคำที่ค่า successor variety เพิ่มขึ้นอย่างรวดเร็ว
     (หลังจาก prefix แล้ว มีตัวอักษรอะไรข้างที่ต่อได้)
  3) Table Lookup ⇒

Term	Stem
engineering	engineer
engineered	engineer
engineer	engineer

4) n-gram : จากคำที่ให้มา เอามาแจกแจงเป็นกลุ่ม กลุ่มละ n ตัวอักษร
- นิยม 2-gram มากสุด (n=2)
- การคำนวนความใกล้เคียงของคำใช้

$$S = \frac{2C}{A+B}$$

A = # diagram ที่แจกแจงคำแรก (ไม่ซ้ำ)
B = # diagram ที่ไม่ซ้ำ ที่แจกแจงจากคำที่สอง
C = # diagram ที่ไม่ซ้ำ ที่ share ร่วมกันระหว่างคำที่ 1 กับ 2

- Q-gram stemmer
แบ่งเป็นกลุ่มละ Q ตัว นับช่องว่าง หน้า, หลังด้วย
แล้วนับว่า คำ 2 คำ มันต่างกันกี่คู่ คำนั้นคือ Q-gram distance

- Index Terms Selection
- ใช้คำนาม
- ถ้าใช้ คำ เป็น index term จะมีปัญหาที่ บางคำรวบแร้ว ความหมายเปลี่ยน ดังนั้นให้ผสมกันระหว่าง word กับ phrase

• Indexing

- เทคนิค
  - Inverted file : ประกอบด้วย
    - vocabulary = คำที่วนใจ (keyword)
    - occurrence = เป็นคำของคำๆ นั้น เช่น ต.น. ที่ปรากฏ, อยู่ใน doc ไหน
  - Suffix array
  - Signature file

- นิยาม - keyword = term ที่ใช้อธิบาย document, สามารถดึงออกมาจาก doc นั่นเองได้
- controlled vocabulary = list ของคำ ที่ใช้เป็น keyword ได้

Inverted File

- มีทั้งบอกตำแหน่งของคำ และอยู่ใน Doc ไหน
- Block addressing คือ แบ่ง text เป็น blocks แล้วหาแหน่งของคำ ก็บอกว่า อยู่ block ไหน
- เหมาะกับระบบที่ Doc ไม่ค่อยเปลี่ยนแปลง (semi-static)

Suffix Tree and Suffix Array

- ทำเป็น tree ก่อน โดยเรียงลำดับตามตัวอักษร จากหน้าไปหลัง
- แล้วทำ suffix array โดยเรียงตำแหน่งของคำ ตามตัวอักษร

Signature Files

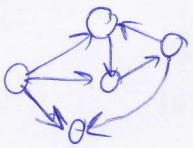
- แบ่ง text เป็น blocks ละเท่าๆกัน
- เอา keyword ที่อยู่ใน text แต่ละคำ ไปหา hash โดยผ่าน signature function ได้ค่า binary ออกมา
- เอา keyword ใน แต่ละ block มาหาค่า hash ถ้า block นั้น มี keyword > 1 ตัว ให้เอาค่า hash มา XOR กัน เอามาเป็นค่าประจำ block
- ทำให้ทุกๆ block จะได้ text signature
- การหาคำที่อยู่ใน block ทำโดย เอาค่า hash ของคำนั้น มา AND กับ ค่า hash ของ block ถ้าเหมือนค่า hash เดิม แสดงว่ามีคำนั้นอยู่

• Multimedia IR

- ข้อมูลมีโครงสร้างซับซ้อน (ภาพ, เสียง)
- Data Modeling
  - RDBMS → การใช้ข้อมูล multimedia ใช้ยาก
  - OODBMS → ใช้คอนเซป object oriented (OO) ← inheritance, polymorphism, encapsulation
    แต่อ่า ประสิทธิภาพไม่ค่อยดี, ยังไม่มาตรฐาน
    → ใช้ภาษา OQL
  - Object-Relational → เก็บข้อมูลเป็นตาราง
    → ใช้ภาษา SQL3 แต่ การจัดการ เป็นแบบ object
- โครงสร้างข้อมูลที่ใช้เก็บข้อมูล ของ DBMS จะใช้ B-tree
- R-tree (Region tree)
  - แทนที่ spatial object ด้วย minimum bounding rectangle (MBR) เป็นกรอบ □ ล้อม object
  - กรอบ MBR ซ้อนกันได้
  - การเก็บเป็น tree เริ่มจาก MBR ใหญ่ แล้วแตก เป็นลูก ก็คือ MBR ที่อยู่ข้างใน
  - ข้อมูลใน tree จะเป็น พื้นที่ ส่วน leaf จะเป็นข้อมูลจริง

Searching The Web

- ของ web site เป็น graph เชื่อมกันไปมา



- มี spider (crawler) ไปเยี่ยม node (page) ต่างๆ เพื่อเอา index กลับมา
- search engine

1) Centerlized Architecture

- crawler-indexer
- Crawler เป็น โปรแกรม ที่ไปเยี่ยม web ต่างๆ ส่ง index กลับมาให้ server กลาง
- ชื่ออื่นของ crawler = robot, spider, wanderer, knowbot
- การ query จะเกิดขึ้นที่ server กลาง โดยใช้ index ที่ crawler ไปหามา
- การสั่งของ crawler จะเป็น Depth-first หรือ Breadth-first Crawling
- มีปัญหาตรงที่ ปริมาณของ ที่ติดต่อกับ server มีมากเกิน

2) Distributed Architecture

- มี 2 ส่วนคือ
  1. Gatherers : คอยเก็บ รวบรวม index จาก web server เอามาส่งให้ Broker
  2. Brokers : เป็นหน้าหาข้อมูลให้ user โดยมันจะเอาข้อมูลมาจาก Gatherer หรือ ถ้ามันก็ไม่รู้ Broker ก็จะไปหาจาก Broker ตัวอื่น
- ปริมาณ load จะไม่สูงเกินไป
- สามารถทำ cache ไว้เก็บคำตอบที่เคยตอบแล้วได้

Information Storage and Retrieval

1 / 2552

1. อธิบายความหมาย

- knowbot คือ โปรแกรมที่ไปเยี่ยม web pages ต่างๆ แล้วส่ง index กลับมายัง server กลาง เพื่อใช้ในการทำ search engine (ชื่ออีกหลายชื่อของ knowbot คือ crawler, robot, spider, wanderer)

- n-gram คือ การแบ่งคำออกเป็นชุด ชุดละ n ตัวอักษรเช่น statistics แจกแจงออกมาเป็น st,ta,at,ti,is,st,ti,cs เมื่อแบ่งเป็น 2-gram (n = 2)

- R-Tree คือ data structure รูปแบบหนึ่งที่ใช้เก็บ spatial object ด้วย minimum bounding rectangle (MBR) ซึ่งเป็นกรอบสี่เหลี่ยมล้อม object ที่เป็นตัวข้อมูล

- Vector Model คือหนึ่งโมเดลในการสืบค้นข้อมูลแบบคลาสสิก โดยแทนที่เอกสาร และ query ด้วยเวกเตอร์ โดยค่าแต่ละมิติในเวกเตอร์คือน้ำหนักแต่มีต่อ keyword แต่ละคำ ความตรงประเด็นของเอกสารวัดจากมุมที่เอกสารทำกับ query โดยยิ่งมุมเล็กยิ่งตรงประเด็น

- Probabilistic Model คือ หนึ่งโมเดลในการสืบค้นข้อมูลแบบคลาสสิก โดยใช้ความน่าจะเป็นที่ผู้ใช้จะเจอเอกสารที่ตรงประเด็น และมีการรับ feedback จากผู้ใช้ว่าเอกสารไหนตรง/ไม่ตรงประเด็น เพื่อปรับความน่าจะเป็น เพื่อให้ครั้งต่อไปสืบค้นเจอเอกสารที่ตรงประเด็นมากยิ่งขึ้น

- Proximity เป็นวิธีการทำ context query คือการหาคำหลายคำ โดยคำนั้นอยู่ห่างกันได้ โดยจะต้องกำหนดระยะห่างสูงสุดระหว่างคำนั้นก่อนเช่นหา (enhance,retrival,4) จะ match “enhance the power of retrieval”

- Supra - index คือ index ที่สร้างขึ้นเหนือ suffix array เพื่อแบ่ง suffix array ออกเป็นช่วงๆ เวลาที่หาคำไหนก็ให้ไปหาใน Supra - index ก่อน ว่าคำนั้นน่าจะอยู่ในช่วงไหนของ suffix array แล้วค่อยไปหาตำแหน่งของคำใน suffix array

- MIMD

- Recall คือ ความสามารถในการดึง doc ที่ตรงประเด็นออกมา มีค่าเท่ากับ จำนวนเอกสารที่ออกมาแล้วตรงประเด็น / จำนวนเอกสารทั้งหมดที่ตรงประเด็น

ค่านี้ใช้วัดประสิทธิภาพของ algorithm ในการ query

- Global Analysis เป็นวิธีการปรับปรุงตัว query แบบอัตโนมัติ โดยใช้ข้อมูลจากเอกสารทั้งหมดในระบบเอามาทำ similarity thesaurus แล้วหาว่าควรเพิ่ม keyword อื่นหรือไม่ และควรปรับ weight เป็นเท่าไหร่

2. ให้เอกสารมา 7 เอกสาร พร้อม query คำนวณหาน้ำหนักของ keyword ของแต่ละเอกสารและน้ำหนักของ keyword ใน query พร้อมจัดลำดับเอกสารที่ถูก return โดยการคำนวณให้คำนึงถึงการเป็นรากศัพท์ด้วย

- ให้ใช้วิธี tf idf ในการหาน้ำหนัก โดยตอนที่หาความถี่ของ keyword ให้คิดถึงรากศัพท์ด้วยเช่นพวกเติม -ed -ing -s -es ถือเป็นคำเดียวกัน

- ส่วนการ query และจัดลำดับให้ทำแบบ vector model จะดีสุด (คิดเป็นเวกเตอร์ ค่าความเหมือนคือ $\cos(\theta)$)

3. กำหนดเอกสารที่ตรงประเด็นทั้งหมดในระบบมาให้ มี 2 อัลกอริทึมในการ query พร้อมผลลัพธ์ที่ได้จากการ query ให้วิเคราะห์โดยใช้หลักการใน Evaluation ทุกวิธี

- ค่าที่ใช้วัดก็มี Average Precision, R-Precision, F-Measure, E-Measure

4. เขียนอัลกอริทึม Proximity query ที่ยอมให้มีระยะห่างระหว่าง keyword ในเอกสารห่างกันได้สูงสุด 6 คำ (ปรับเปลี่ยนจาก phrasal search) พร้อมยกตัวอย่างประกอบ

- หา doc ที่มี keyword แต่ละคำใน phrase ทุกคำ

- แต่ละ doc ที่มี keyword อยู่ให้สร้าง array เก็บตำแหน่งของคำแต่ละคำใน phrase

- เริ่มจากคำที่ขนาด array น้อยที่สุด

- วนลูปแต่ละตำแหน่งใน array แล้วไปหา array ของคำอื่นๆ ในตำแหน่งที่อยู่ใกล้กับตำแหน่งนั้นที่สุด แล้วดูว่าระยะห่าง keyword ห่างเกิน 6 หรือไม่ ถ้าไม่เกินแสดงว่าเอกสารนั้นเป็นคำตอบ

- ถ้าเกินให้ดูตำแหน่งต่อไปจนกว่าจะเจอ หรือวนครบทุกตำแหน่งแล้วไม่เจอก็แปลว่าเอกสารนั้นไม่ใช่คำตอบ

5. มีเอกสารมาให้ 9 เอกสารพร้อม keyword ในเอกสาร กำหนดให้มี 3 processors ให้แสดงวิธีการแบ่งการทำงานแต่ละแบบพร้อมวิเคราะห์