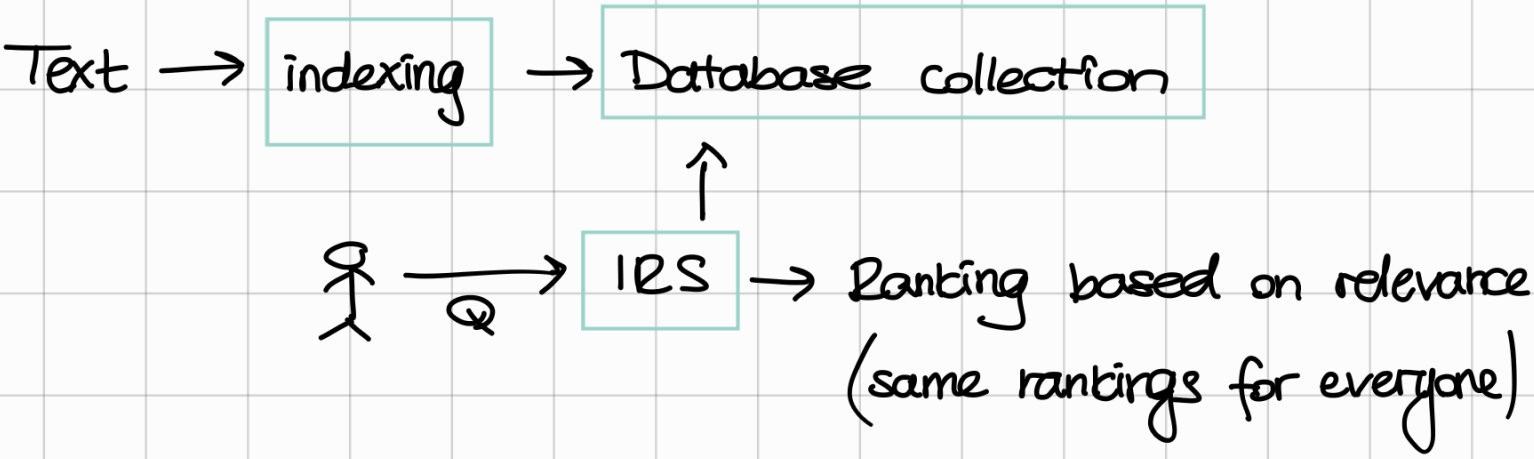
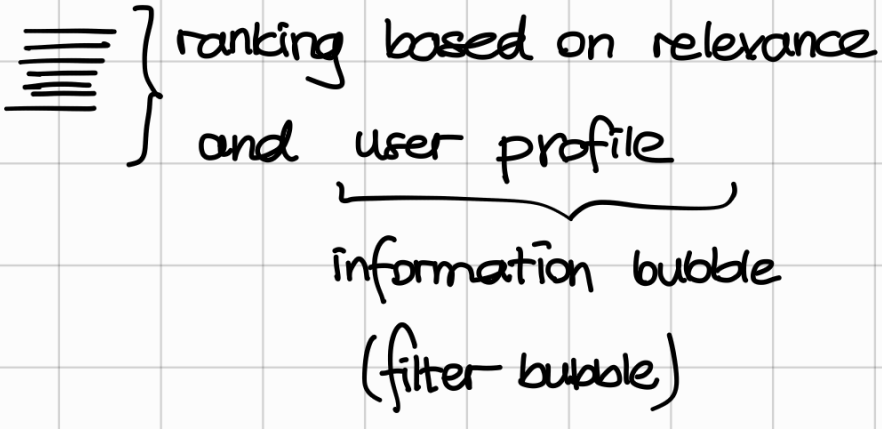
22.09.2023

Information Systems

- Database Systems (structured)
- Info. Retrieval Systems (unstructured) (text)
- Generative AI based Systems (text)





Words → Indexing terms

Root / stem (root of the word)

(approx. root

compute / comput
root stem)

e.g. comput-
for computation
computer

indexing:

- determines words (terms) to be used for the description of documents.
- assigns importance (weights) to terms.

Program: **Porter's stemming algorithm**

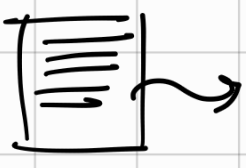
Lemmatization: indexing w/o using the orig. word

e.g. good, better, best

e.g. Zemberek

Stopwords: ∃ stopword lists. In those lists there are commonly used words (function words).

e.g. bir, ve, bu, ki, mi, sonra ....



30-40 % stopwords

50 % very infreq. words

Text

w_1 w_2 w_3 w_1 w_2 w_3 (w: word)

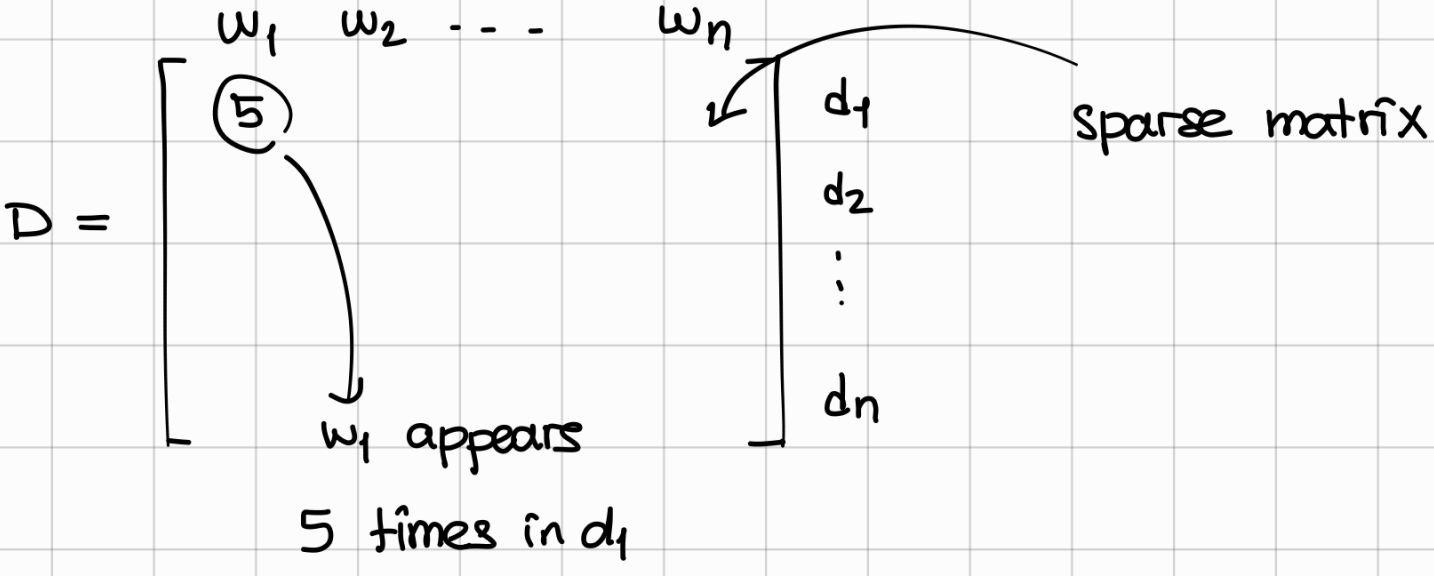
Doc 1

Doc 2

⋮

Doc n

Document Term Matrix



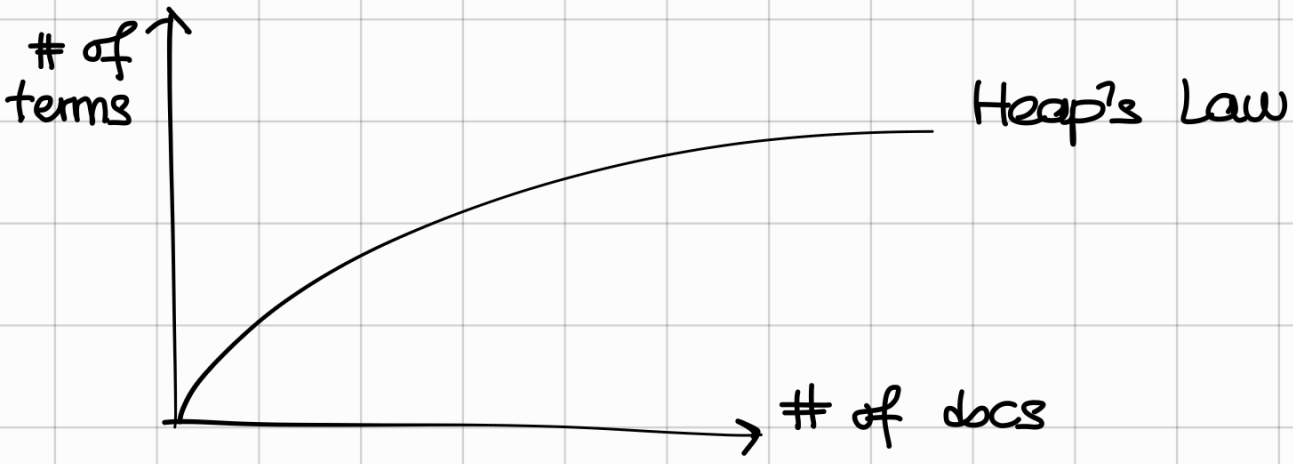
TF-IDF: term freq. – inverse document freq.

Gerard Salton & Chris Buckley ?

Term weight assignment

Medical Collections

MEDLARS, MEDLINE



Heap's Law: # of docs ↑, # of terms ↑
(but at a slower rate)

Zipf's Law: freq. of word $\propto \frac{1}{\text{rank of word}}$

IR System **Evaluation**

(evaluate based on)

Effectiveness & Efficiency

Relevant docs time memory

Measuring Performance

- **Online**: User satisfaction
- **Lab**: Eval is based on test collection

Test Collection ⊃

- A doc. set (collection)
- A query set (query collection)
- Relevent docs / queries

TREC (Text Retrieval Conf.)

Pooling

Group 1 $q_1 \rightarrow d_1, d_5, d_7$

Group 2 $q_1 \rightarrow d_5, d_{10}$

↳ $q_1 \rightarrow d_1, d_5, d_7, d_{10}$

Fleiss Kappa Measure to judge how dependable the decisions are.

- Annotator, assessor
- Limitation of pooling: There might be some docs that are not shown to any assessor & assumed irrelevant.

treceval:

"TRECeval" refers to the software tool used for evaluating the performance of information retrieval systems in the context of TREC. It provides the means to compute various evaluation metrics, such as precision, recall, F1 score, and mean average precision (MAP), which are commonly used to assess the quality of retrieval results.

26.09.2023

Data set

Doc. collection

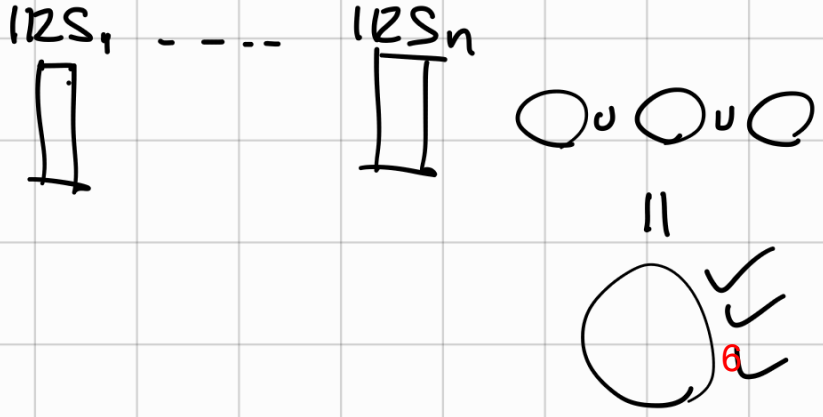
query set

Rel. docs for each query

- Use test collections for reproducibility.

Pooling → TREC

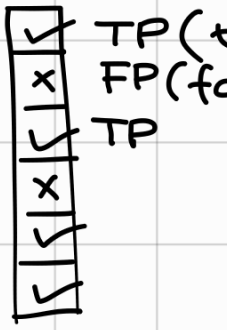
↙



Pooling in information retrieval systems is a process used to combine and rank the results obtained from multiple retrieval models or algorithms in order to generate a final ranked list of documents for a given query.

Effectiveness Measures

- Precision, recall, f measure
- positive: deemed relevant by IRS



TN
FN
(false neg: rel. docs that are not shown in / retrieved by the system)

$$\frac{4}{6} = \frac{\#\text{ rel. docs}}{\#\text{ docs we observed}}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

R @ 10

Prec @ 5
Prec @ 10
Prec @ 15
→ # of docs

	IRS 1	IRS 2
q_1	0.5	0.7
⋮		
q_{50}	0.3	0.4
Avg	0.75	0.7

paired t-test

p=0.05

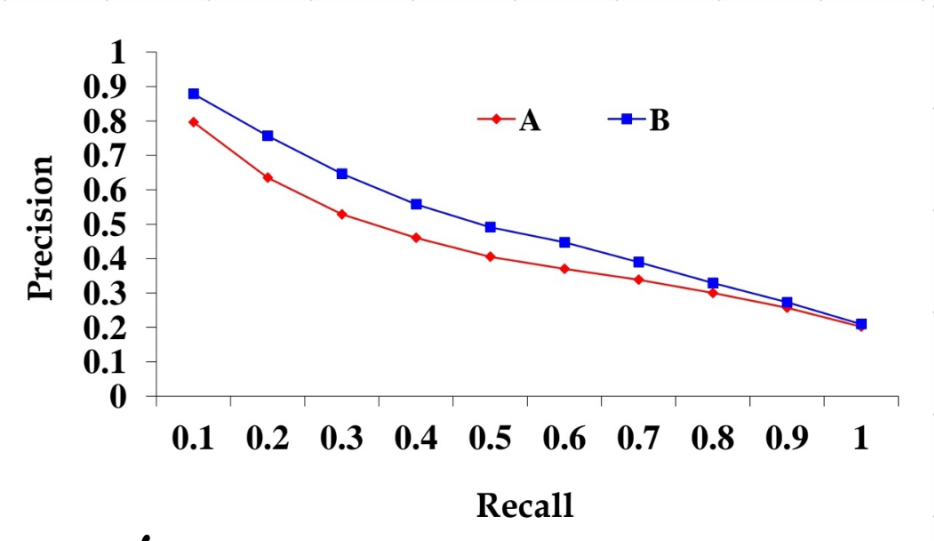
ANOVA

TODO. check ANOVA & MONOVA

$F = \frac{2PR}{P+R}$ P: precision, R: recall

Rank	1	2	3	4	5	6	7	8	9	10
Relevance	0	1	0	1	1	1	1	0	0	0
Precision	0/1	1/2	1/3	2/4	3/5	4/6	5/7	5/8	5/9	5/10
Recall	0/10	1/10	1/10	2/10	3/10	4/10	5/10	5/10	5/10	5/10

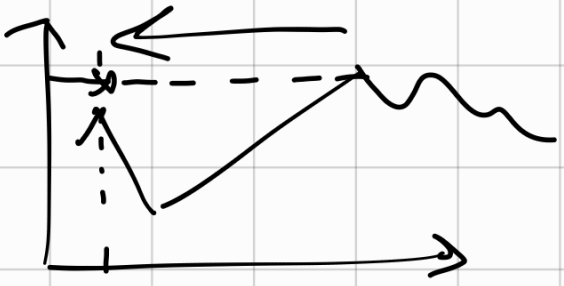
rel. docs = 10



↳ B is better than A.

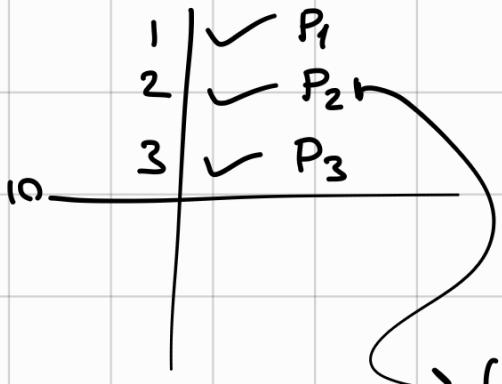
Interpolation of Prec. Values

- The highest prec. val. after that we call pts is used as the prec. val..



MAP (Mean Avg Precision)

- We know # rel. docs



→ (not necessarily the second doc. might be 7th doc.)

$$MAP_{10} = \frac{P_1 + P_2 + P_3}{\#\text{ rel. docs}}$$

↓ the pos. where we cut

$$MAP_n = \frac{\sum_i P_i \quad st \quad i \in \{\text{rel. docs}\} \cap \{\text{top } n \text{ docs}\}}{\#\text{ rel. docs } \sout{\text{in top } n \text{ docs}}}$$

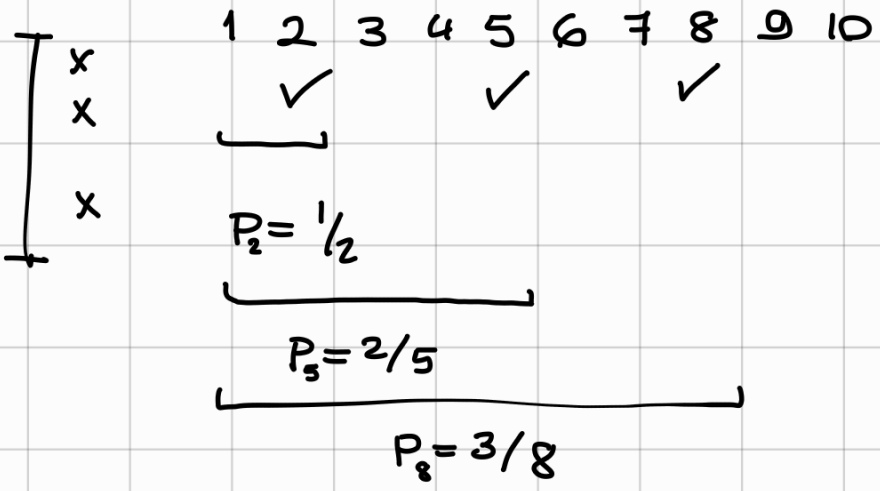
29.09.2023

- There are diff. defs of MAP.

We know the rel. docs, we haven't seen all of them.

Consider top 10 (ranked) docs

Assume that ∃ 5 rel. docs (3 rel. doc. in top 10)



$$MAP_{10} = \frac{P_2 + P_5 + P_8}{\#\text{ rel. docs for this query}}$$

$$= \frac{0.5 + 0.4 + 0.375}{5}$$

Performance of a System

Q No	MAP_{10}
1	...
2	...
3	...
Avg	

or

Q No	MAP (X)
1	MAP_5
2	MAP_{10}
⋮	⋮
Avg	

→ X: # rel. docs for a particular query

Raghavan Schütze

- Info. Ret. - Schütze, Search Engines - Croft, Stohlman
  - " " - van Rijsbergen
    - ↳ ∃ evaluation dupt.

03.10.2023

- w/ 1000000 doc.s we might use only 100 docs. ⇒ limited accumulators
- Paper: Inverted Index Structures for Search Engines by Zohel, Moffad.

Similarity Calculations (among the docs of a collection)

$$D = \begin{bmatrix} t_1 \dots t_n & \\ & d_1 \\ & \vdots \\ \text{sparse} & \\ & d_m \end{bmatrix} \qquad S = \begin{bmatrix} 1 & s_{12} & s_{13} & s_{14} \\ s_{21} & 1 & s_{23} & s_{24} \\ s_{31} & s_{32} & 1 & s_{34} \\ s_{41} & s_{42} & s_{43} & 1 \end{bmatrix}$$

↓ doc.term matrix

similarity matrix

$s(d_i, d_i) = 1$

$s_{ij} = s_{ji}$ (symmetric) (there can be an asym. formula as well)

- S can be used in clustering docs

$$\begin{bmatrix} 1 & s_{12} & s_{13} & s_{14} \\ s_{21} & 1 & s_{23} & s_{24} \\ s_{31} & s_{32} & 1 & s_{34} \\ s_{41} & s_{42} & s_{43} & 1 \end{bmatrix}$$

m-1 docs

$\frac{m.(m-1)}{2}$ calculations

- Matrix D is weighted / binary
  - Dice coefficient
  - Cosine similarity
  - Dot product
  - Jaccard
  ⋮

Similarity Coef.	Binary	Weighted
Dot product (Inner product)	$X \cap Y$	$\sum X_i Y_i$
Cosine	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert^{1/2} \cdot \lvert Y \rvert^{1/2}}$	$\frac{\sum X_i Y_i}{\sqrt{\sum X_i^2 \sum Y_i^2}}$
Dice	$\frac{2\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$	$\frac{2\sum X_i Y_i}{\sum X_i^2 + \sum Y_i^2}$
Jaccard	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert - \lvert X \cap Y \rvert}$	$\frac{\sum X_i Y_i}{\sum X_i^2 + \sum Y_i^2 - \sum X_i Y_i}$

(i: index of the i^{th} term)

Example

$X = (1\ 0\ 1\ 1\ 1)$ $|X| = 4$

$Y = (1\ 1\ 0\ 1\ 0)$ $|Y| = 3$

2 common items $= X \cap Y$

Dot product 2

Cosine $1/\sqrt{3}$

Dice $4/7$

Jaccard $\frac{2}{3+4-2} = \frac{2}{5}$

Example

$X = (2\ 0\ 1\ 3\ 2)$

$Y = (1\ 0\ 2\ 1\ 5)$

$$\text{Dice} = \frac{2(2+2+3+10)}{(4+1+9+4)+(1+0+4+1+25)} = 0.69$$

Cosine $\cong 0.72$

- When calculating the S matrix:

1. Brute force approach

# of entries to be calculated $= \frac{m.(m-1)}{2}$

2. Use the knowledge of term dist. in doc

$$D = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix}$$

$m = 5$ # of docs

$n = 6$ # of terms

↳ ∃ at least one "1" in each row & col.

represent D like this instead:

$t_1 \rightarrow d_1, d_2$ $\quad$ $t_4 \rightarrow d_2, d_5$

$t_2 \rightarrow d_1, d_2$ $\quad$ $t_5 \rightarrow d_1, d_2$

$t_3 \rightarrow d_4, d_5$ $\quad$ $t_6 \rightarrow d_3, d_4, d_5$

Consider d_1: ∃ t_1, t_2, t_5.

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \cup \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \cup \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$$

→ so only d_2 can have non-zero similarity with d_1.

Calculate S_{12}.

Similarly for d_2: $\{d_1, d_2, d_5\} \Rightarrow$ Calc. S_{25}.

d_3: $\{d_3, d_4, d_5\} \Rightarrow$ " S_{34}, S_{35}.

d_4: $\{$ " " $\} \Rightarrow$ " S_{45}.

d_5: $\{d_2, d_3, d_4, d_5\} \Rightarrow S_{25}, S_{35}, S_{45}$ are already calculated.

5 calcs = $S_{12}, S_{25}, S_{34}, S_{35}, S_{45}$.

vs

$\frac{4.5}{2} = 10$ calcs bruteforce.

06.10.2023

Continuing from last lecture's example:

- x_d = depth of indexing (avg. # of terms / doc)
- t_g = term generability (avg. # of docs / term)

$$D = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix}$$

3. Using the inverted index file:

$d_1 \rightarrow t_1, t_2, t_5$

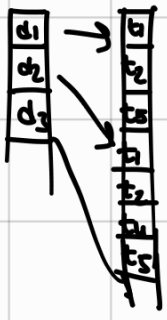
$d_2 \rightarrow t_1, t_2, t_4, t_5$

Inverted Index

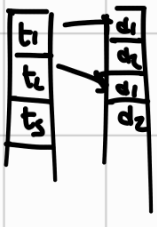
$t_1 \rightarrow d_1, d_2$ $\quad t_4 \rightarrow d_2, d_5$

$t_2 \rightarrow d_1, d_2$ $\quad t_5 \rightarrow d_1, d_2$

$t_3 \rightarrow d_4, d_5$ $\quad t_6 \rightarrow d_3, d_4, d_5$



Inverted file:



posting list

$$\text{Dice coef.} = \frac{2|X \cap Y|}{|X| + |Y|}$$

Document length array:

d_1	d_2	d_3	d_4	d_5
3	4	1	2	3

Consider d_1.

similarity array:

s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
x	ø	0	0	0

d_1 contains these terms:

t_1 x

t_2 ø

t_5 3

$$\Rightarrow s_{12} = \frac{2.3}{3+4}$$

$$s_{13} = s_{14} = s_{15} = 0.$$

Similarly for d_2.

s_{21}	s_{22}	s_{23}	s_{24}	s_{25}
x	x	0	0	ø

t_1 - - -

t_2 - - -

t_4 1

t_5 - - -

$$\Rightarrow s_{22} = s_{24} = 0.$$

$$s_{25} = \frac{2.1}{4+3}$$

Similarly for d_3.

x	x	x	ø	ø

t_6 1 1

$$\Rightarrow s_{34} = \frac{2.1}{1+2}$$

$$s_{35} = \frac{2.1}{1+3}$$

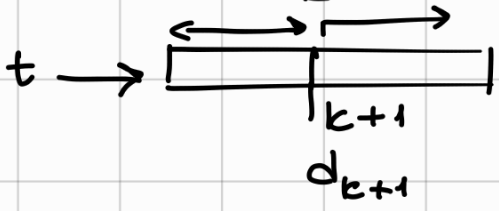
- Consider m doc.s

$\forall$ doc. consider X_d # terms

$\forall$ term consider t_g # doc.s

$$O(m \cdot X_d \cdot t_g)$$

Consider d_k. Consider a term t of this doc.



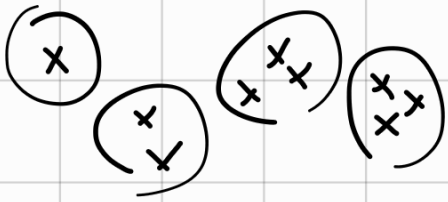
for d_1, only ignore d_1 — for d_2, ignore d_1 and d_2

$$= \overbrace{X_d \cdot t_g \cdot \frac{m-1}{m}}^{d_1} + \overbrace{X_d \cdot t_g \cdot \frac{m-2}{m}} + \dots + X_d \cdot t_g \cdot \frac{1}{m}$$

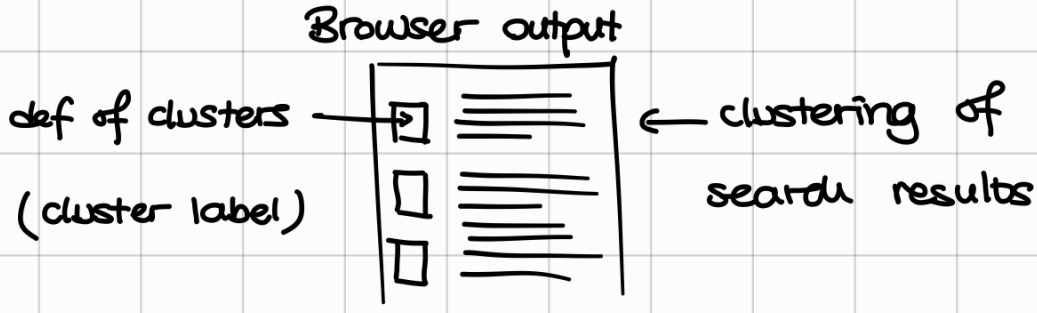
$$= X_d \cdot t_g \cdot \frac{(m-1)\cancel{m}}{2} \cdot \frac{1}{\cancel{m}} = X_d \cdot t_g \cdot \frac{(m-1)}{2} = O(X_d t_g m)$$

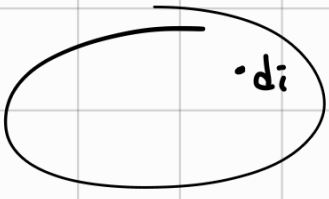
- Difference between 2 and 3 is we calculate $|X \cap Y|$ (of all the docs that share terms with d_i) while calculating the similarity array of d_i.

Clustering



- How to use clustering for information retrieval?





← docs similar to d_i.
(we can obtain additional rel. docs)

Cluster Based Retrieval

Identify best matching clusters

Implementation:

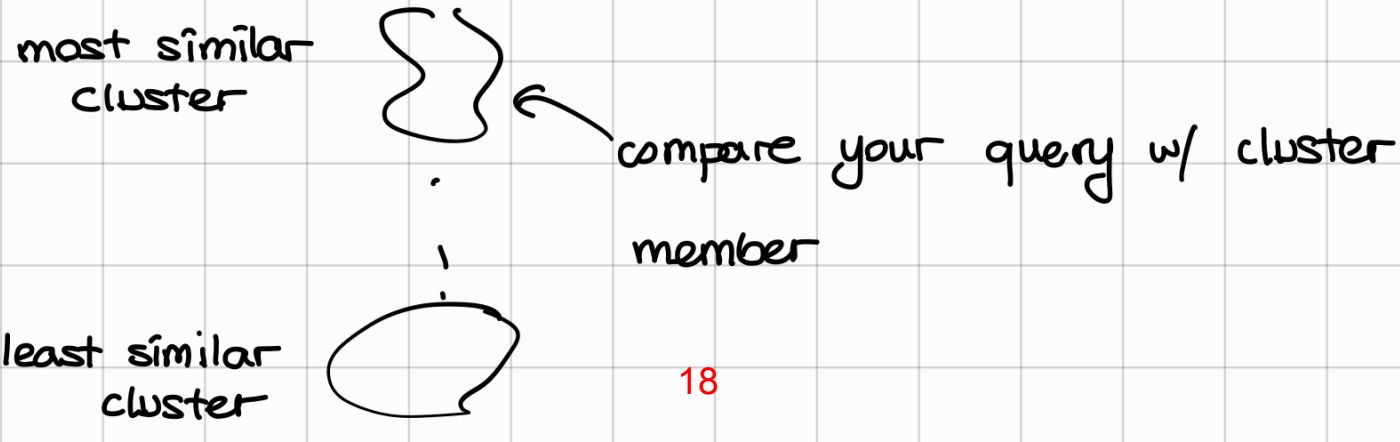
$\forall$ cluster obtain a centroid ←

$$C = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1+1}{2} & 0 & \frac{1+1}{2} & \frac{1+0}{2} & 0 \end{bmatrix}$$

$$= [1 \; 0 \; 1 \; 0.5 \; 0]$$

Compare query w/ centroid

Rank clusters according to their similarity to the query

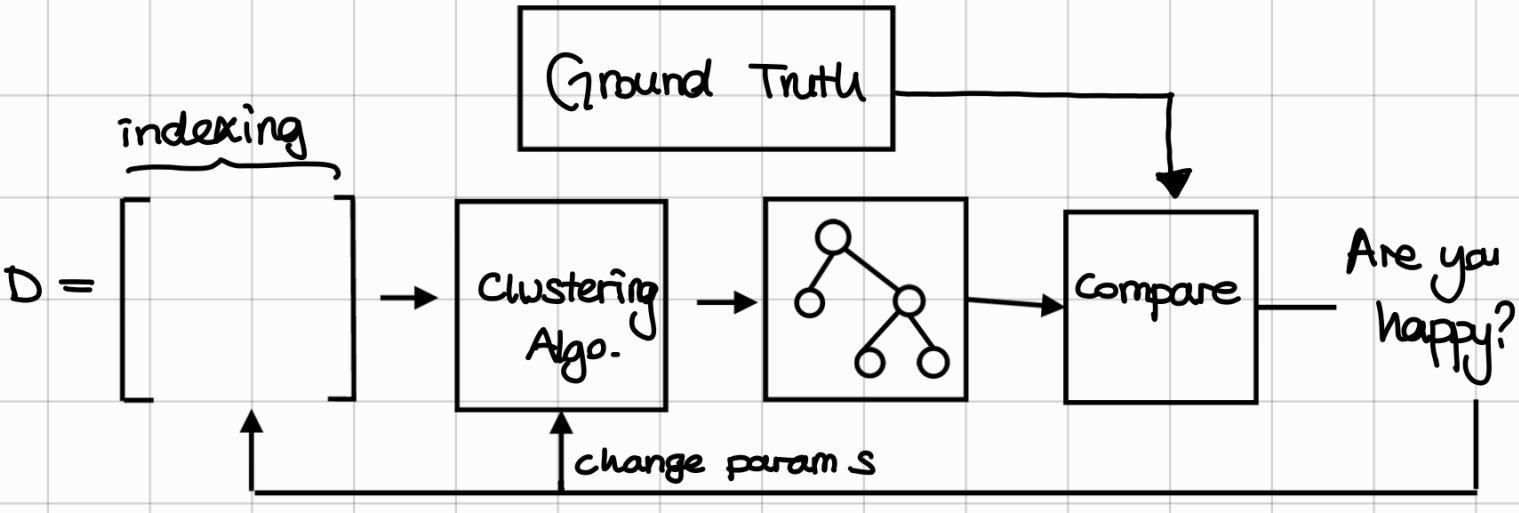


- Classification : (classes are known) supervised
- Clustering : unsupervised
  ↳ after clustering we need to label (give summary tag) to clusters.

Cluster Hypothesis:

Docs rel. to the same query would appear in the same cluster.

- Tom Hope, Inflection in Scientific Discoveries.



Robustness small changes in D matrix shouldn't change clusters much

Classifying Clustering Algorithms

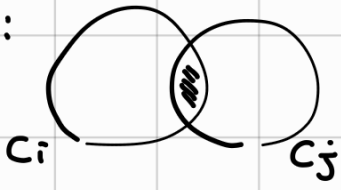
- **Structure:**

According to the structure generated:

1. Partitioning: clusters do not have common items.

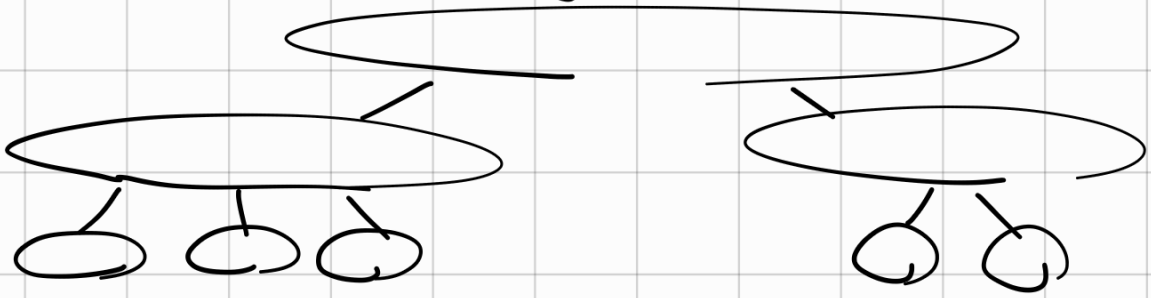
$$C_i \cap C_j = \emptyset \quad \forall\, i \neq j.$$

2. Overlapping:

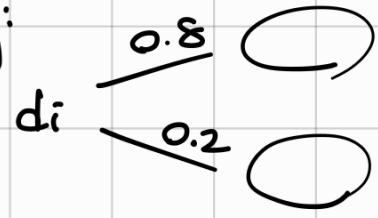


$C_i \cap C_j \neq \emptyset$

3. Hierarchical clustering:



4. Fuzzy clustering:



10.10.2023

- **Procedure**

According to how the clusters are generated:

1. Single-pass:

each item is considered only once.

- select some data item as cluster initiator (seed)

Assign non-seeds to these clusters

↳ hand generated v select some of them? v random

2. Multi-pass:

obtain initial clusters. then try to improve them.

3. Heuristic algo.s:

First doc inits a cluster

Consider next one

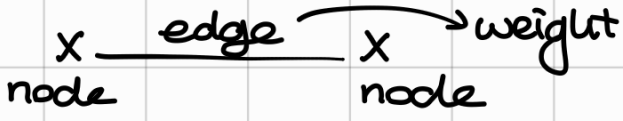
If it's diff. it starts its own cluster

-order dependent

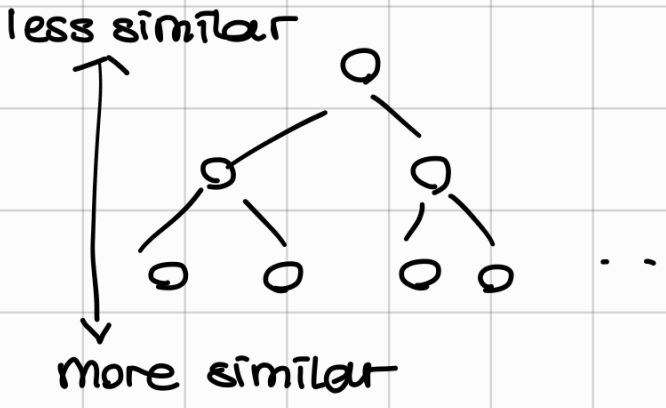
-good for new event detection and tracking

4. Hierarchical:

- Graph theoretical:



- single link
- complete link
- avg. link

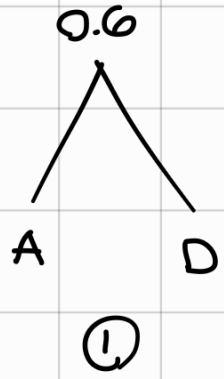


. .

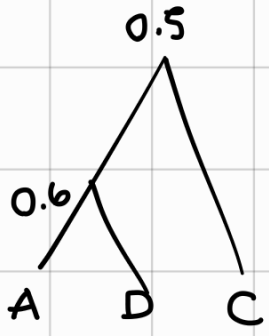
$$S = \begin{array}{c} \begin{matrix} A & B & C & D \end{matrix} \\ \begin{bmatrix} 1.0 & 0.3 & 0.5 & 0.6 \\ & 1.0 & 0.4 & 0.5 \\ & & 1.0 & 0.3 \\ & & & 1.0 \end{bmatrix} \end{array} \begin{matrix} A \\ B \\ C \\ D \end{matrix}$$

i. Single-link: — not order dependent

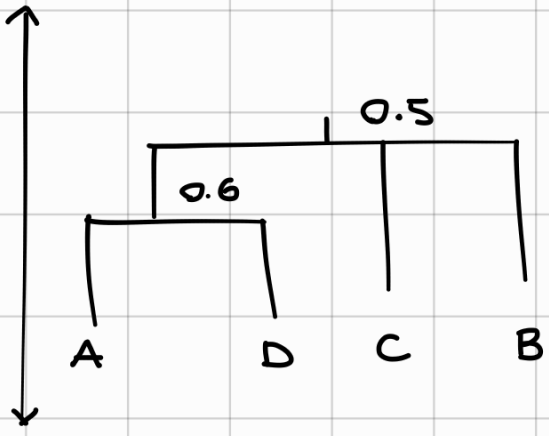
step	pair	sim. value
1	AD	0.6
2	AC	0.5
3	BD	0.5
4	BC	0.4
5	AB	0.3
6	CD	0.3

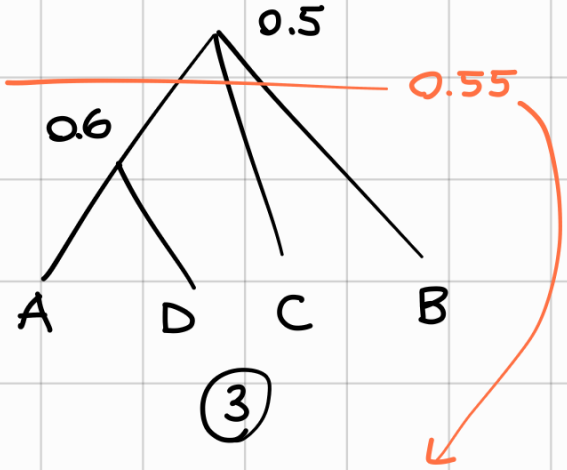


①



②





partitioning threshold

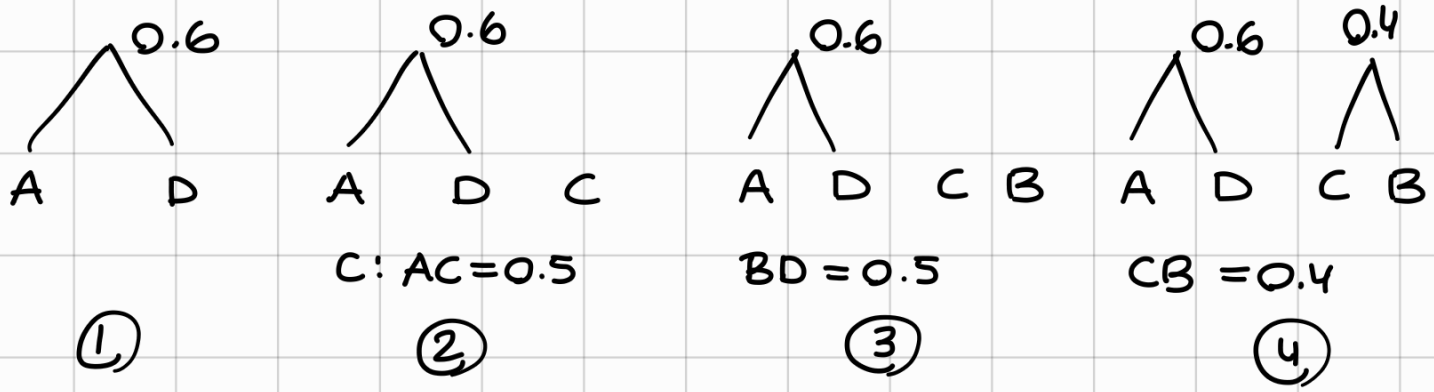
(AD) (C) (B)

Cluster Validation:

$$S' = \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 1.0 & 0.5 & 0.5 & 0.6 \\ & 1.0 & 0.5 & 0.5 \\ & & 1.0 & 0.5 \\ & & & 1.0 \end{bmatrix} \end{array} \begin{array}{c} A \\ B \\ C \\ D \end{array}$$

(single-link: the similarity between a cluster and new item is taken as the most similar member of that cluster for that item.)

ii. Complete Link:



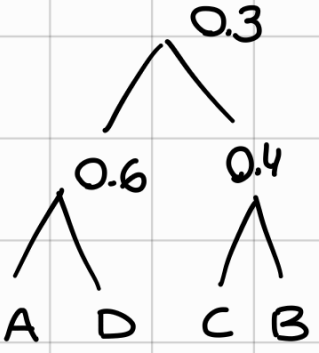
① ② ③ ④

- AD 0.6
- AC 0.5
- BD 0.5
- BC 0.4

AB 0.3] intercluster

CD 0.3] intercluster

⑤

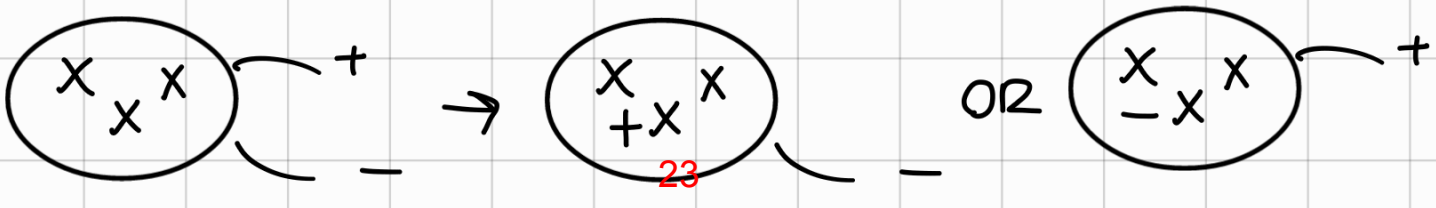


min { AC, AB, DC, DB }

= { 0.5, 0.3, 0.3, 0.5 }

= 0.3

- final clustering structure is order dependent



Characteristics of Desirable Clustering Algorithms

1. Effectiveness & Efficiency - time
   - meaningful clusters
   - memory

2. Maintainable: we prefer an incremental approach
   - shouldn't compute from scratch w/ each new data)
   - reclustering vs incremental clustering

3. Order independent

4. Robustness: small error in data shouldn't effect the clustering structure

5. Addition of new data shouldn't destroy the existing structure

6. Cluster sizes are comparable (partitioning)

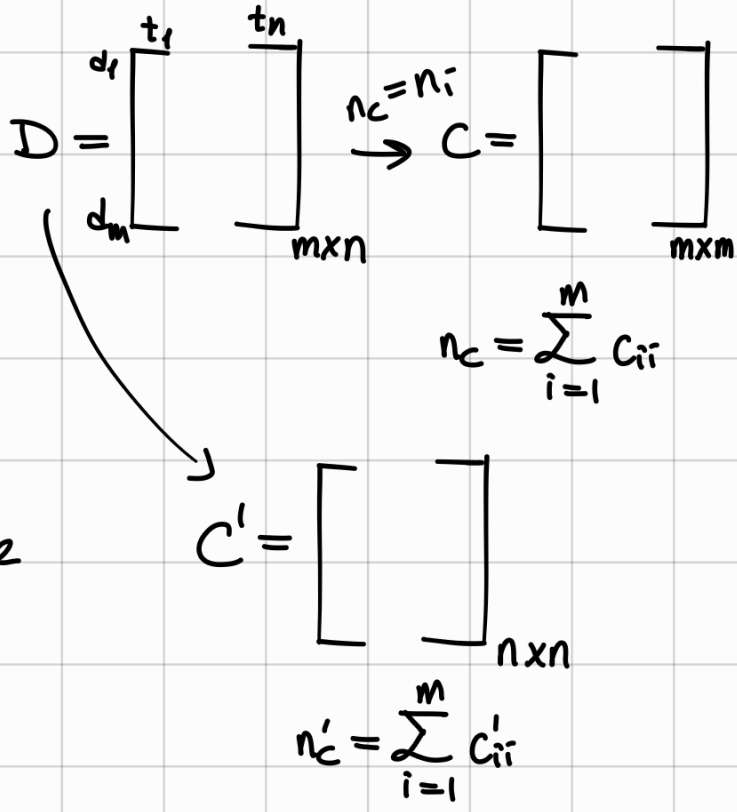
7. Small number of hyperparameters

e.g. Fazlı hoca's paper : Cover Coef.-based Clustering Methodology C^3M

Partitioning

Seed-oriented:

1. Find # clusters
2. Select cluster seeds
3. Assign nonseeds to clusters based on their similarity to the cluster seeds



C : cover coef. matrix

13.10.2023

→ dendogram

dendrogram

Cover Coef.-based Clustering Methodology (C^3M)

$$D = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix} \quad mxn$$

$\alpha_1 = 1/3 \quad \alpha_2 = 1/4 \quad \alpha_3 = 1/3$

$\alpha_4 = 1/2 \quad \alpha_5 = 1/2$

$\beta_1 = 1/4 \quad \beta_2 = 1/1 \quad \beta_3 = 1/2 \quad \beta_4 = 1/2 \quad \beta_5 = 1/2 \quad \beta_6 = 1/3$

$\alpha_i : \frac{1}{\#\text{ terms in } d_i} \qquad \beta_j : \frac{1}{\#\text{ docs term } j \text{ appears in}}$

$$S = \underbrace{\begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{matrix} \\ \begin{bmatrix} \alpha_1 & 0 & 0 & \alpha_1 & 0 & \alpha_1 \\ \alpha_2 & \alpha_2 & \alpha_2 & \alpha_2 & 0 & 0 \\ \alpha_3 & 0 & 0 & 0 & \alpha_3 & \alpha_3 \\ 0 & 0 & 0 & 0 & \alpha_4 & \alpha_4 \\ \alpha_5 & 0 & \alpha_5 & 0 & 0 & 0 \end{bmatrix} \end{array}}_{\alpha} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix} = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{matrix} \\ \begin{bmatrix} 1/3 & 0 & 0 & 1/3 & 0 & 1/3 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix}$$

↳ the same as dividing each elem. in a row by the row sum. $\left(\text{elem} \to \frac{\text{elem}}{\text{row sum}}\right)$

$$S' = \underbrace{\begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{matrix} \\ \begin{bmatrix} \beta_1 & 0 & 0 & \beta_4 & 0 & \beta_6 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & 0 & 0 \\ \beta_1 & 0 & 0 & 0 & \beta_5 & \beta_6 \\ 0 & 0 & 0 & 0 & \beta_5 & \beta_6 \\ \beta_1 & 0 & \beta_3 & 0 & 0 & 0 \end{bmatrix} \end{array}}_{\beta} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix} = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{matrix} \\ \begin{bmatrix} 1/4 & 0 & 0 & 1/2 & 0 & 1/3 \\ 1/4 & 1 & 1/2 & 1/2 & 0 & 0 \\ 1/4 & 0 & 0 & 0 & 1/2 & 1/3 \\ 0 & 0 & 0 & 0 & 1/2 & 1/3 \\ 1/4 & 0 & 1/2 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix}$$

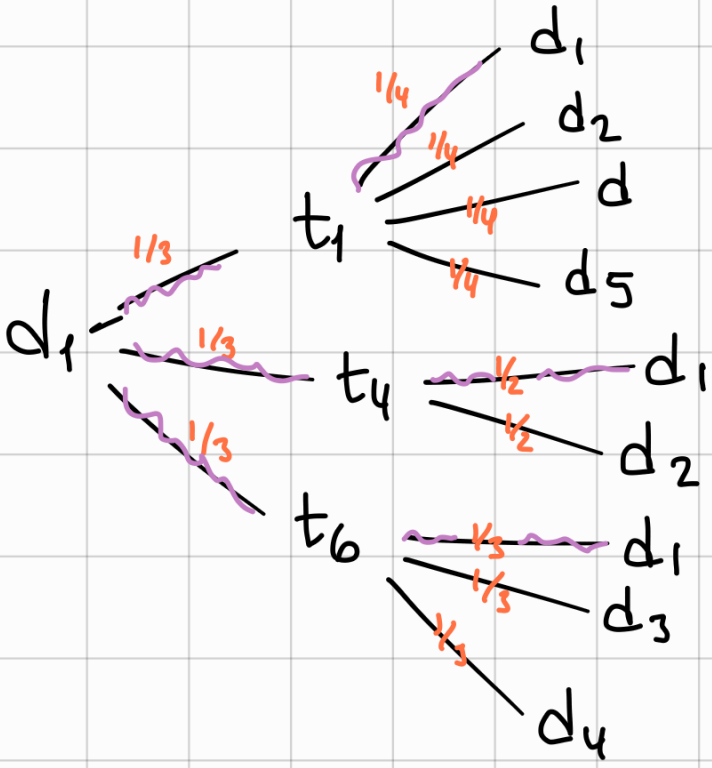
↳ $\left(\text{elem} \to \frac{\text{elem}}{\text{col. sum}}\right)$

$$S'^T = \begin{bmatrix} \beta_1 & \beta_1 & \beta_1 & 0 & \beta_1 \\ 0 & \beta_2 & 0 & 0 & 0 \\ 0 & \beta_3 & 0 & 0 & \beta_3 \\ \beta_4 & \beta_4 & 0 & 0 & 0 \\ 0 & 0 & \beta_5 & \beta_5 & 0 \\ \beta_6 & 0 & \beta_6 & \beta_6 & 0 \end{bmatrix}$$

$$\underset{m\times m}{C} = \underset{m\times n}{S} \cdot \underset{n\times m}{S'^T}$$

m: # docs

$$C = \begin{bmatrix} C_{11} = 1/3 \cdot 1/4 + 0 + 0 + 1/3 \cdot 1/2 + 0 + 1/3 \cdot 1/3 = 0.361 & C_{12} \cdots \\ \vdots & \end{bmatrix}$$



$$C_{ij} = \alpha_i \sum_{k=1}^{n} d_{ik}\, \beta_k\, d_{jk}$$

$$C_{ij} = S_i^{th\ row} \times S'^{T\ th\ col.}_j = \sum_{k=1}^{n} S_{ik} \times S'^{T}_{kj}$$

$$= \sum_{k=1}^{n} (\alpha_i \cdot d_{ik}) \cdot (\beta_k \cdot d_{kj})^T$$

$$= \sum_{k=1}^{n} \alpha_i \cdot d_{ik} \cdot \beta_k \cdot d_{kj}$$

- $C_{ii} \searrow$ if d_i has a small # common terms w/ the rest of the collection.

- $n_c = \sum_{i=1}^{m} c_{ii}$ = # clusters

- If all docs are unique: $d_i \cap d_j = \emptyset \quad \forall\ i \neq j$

$$C = \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix} \qquad n_c = m.$$

- If all docs are identical: $d_i \cap d_j = d_i = d_j \quad \forall\ i, j$

$$C = \begin{bmatrix} 1/m \end{bmatrix} \qquad n_c = 1.$$

- decoupling coef. for doc $i = \delta_i = c_{ii}$.

avg. decoup. coef. $= \delta = \dfrac{\sum c_{ii}}{m} = \dfrac{\sum \delta_i}{m}$

- row i: $c_{ii} = c_{ij}$ if $d_i = d_j$

$c_{ii} \geq c_{ij}$ else

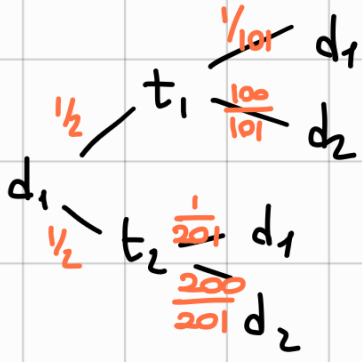
- $c_{ij} = 0 \iff c_{ji} = 0.$

- For a <u>weighted</u> D matrix c_{ii} can be smaller than c_{ij}

e.g. ↓

e.g.

$C_{ii} < C_{ij}$



$$\begin{matrix} d_1 \\ d_2 \end{matrix}\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 100 & 200 & 0 & 0 & 0 \end{bmatrix}$$

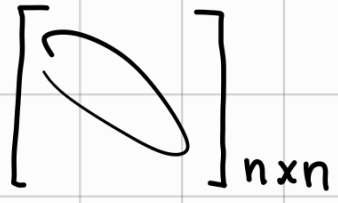
$$C_{11} = \frac{1}{2}\cdot\frac{1}{101} < C_{12} = \frac{1}{2}\cdot\frac{100}{101}$$

$D \rightarrow C' = \begin{bmatrix} & \\ & \end{bmatrix}_{n\times n}$ ← cover coef. matrix for terms C'_{ii}.

$$C'_{ij} = \beta_i \sum_{k=1}^{m} d_{ki}\,\alpha_k\,d_{kj}$$

(Remark.

$$C_{ij} = \alpha_i \sum_{k=1}^{n} d_{ik}\,\beta_k\,d_{jk}$$)

$C' =$  $_{n\times n}$

$$n_{C'} = \sum_{i=1}^{n} C'_{ii} = \sum_{i=1}^{n} \delta'_i$$

↳ decoupling of terms

↓

$n_{C'} = n_C$

$$\delta' = \frac{\sum_{i=1}^{n} \delta_i'}{n} \longrightarrow n_{C'} = \delta' . n \Rightarrow \delta' . n = \delta . m$$

$$n_C = \delta . m$$

$$\Rightarrow \delta' = \delta . \frac{m}{n}$$

$C = [\,\cdot\,]_{m \times m}$ $\quad C' = [\,\cdot\,]_{n \times n}$

$\min(n_c) = 1$

$\max(n_c) = \min(m,n)$

d_c : avg. # docs / cluster

$d_{c'}$: " " terms / "

δ_i = decoupling coef of d_i.

$D \rightarrow S \longrightarrow n_c$

- select seeds (docs)
- assign nonseeds to seeds $(m - n_c)$

$$\begin{pmatrix} \underbrace{(m-n_c)}_{\text{nonseeds}} & \underbrace{n_c}_{\text{seeds}} \end{pmatrix}$$

$\psi_i = 1 - \delta_i$ coupling coef.

- we want high decoupling
  & high <u>seed power</u>.

p_i = seed power of doc i

$= \delta_i \cdot \psi_i \cdot \underbrace{(\#\text{ terms in doc}_i)}_{X_d}$

Remark. X_{di} = avg. # terms per doc.

$p_1 = 0.361 \cdot (1-0.361) \cdot 3 = 0.692$

$p_2 = 0.563 \cdot (1-0.563) \cdot 4 = 0.984$

$p_3 = 0.692$

$p_4 = 0.484$

$p_5 = 0.469$

17.10.2023

decoupling coef

$$C = \begin{bmatrix} 0.361 & 0.250 & 0.194 & 0.111 & 0.083 \\ 0.188 & 0.563 & 0.063 & 0 & 0.188 \\ 0.194 & 0.083 & 0.361 & 0.277 & 0.088 \\ 0.167 & 0 & 0.417 & 0.417 & 0 \\ 0.125 & 0.375 & 0.125 & 0 & 0.375 \end{bmatrix}_{m \times m}$$

$C_{ii} = \delta_i$

$\psi_i = 1 - \delta_i$

$\sum_{j=1}^{m} C_{ij} = 1.$

$C_{ij} = \alpha_i \sum_{k=1}^{n} d_{ik} \beta_k d_{jk}$ $\quad n_c = n_{c'}$

→ d_4 & d_5 have no common term

- All docs are unique: $C = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$ $n_c = m$.

- " " " identical: $C = \begin{bmatrix} 1/m \end{bmatrix}$ $n_c = 1$.

$$n_c = \sum c_{ii} = n_c' = \sum c_{ii}' \qquad C' = \begin{bmatrix} & \\ & \end{bmatrix}_{n \times n}$$

$\underbrace{(m - n_c)}_{\text{non seeds.}} . n_c$

Seed Selection

p_i = seed power of doc i

$= \delta_i \cdot \psi_i \cdot (\#\text{ terms in doc}_i)$

$= c_{ii} \cdot (1 - c_{ii}) \cdot (\#\text{ terms in doc}_i)$

$\underset{c_{ii}}{\text{argmax}}\, p_i = 0.5$

$p_1 = 0.692$ $\quad p_3 = 0.692$ $\quad p_5 = 0.469$

$p_2 = 0.984$ $\quad p_4 = 0.484$

d_2 is a seed — two candidate seeds: d_1 & d_3.

False Seed Elimination

Check if d_1 & d_3 are identical?

(If the difference is less than a threshold, assume they are identical)

$C_{ij} = C_{ji} = C_{ii} = C_{jj} \Rightarrow d_i$ & d_j are identical.

d_1 and d_3 are not identical.

$d_2 \rightarrow t_1, t_2, t_3, t_4$ $\quad$ $d_1 \rightarrow t_1, t_4, t_6$

$d_3 \rightarrow t_1, t_5, t_6$ } Is a seed now bc it has less terms in common w/ d_2.

$$C = \begin{matrix} & & 2 & 3 & \\ & \downarrow & \downarrow & & \\ \end{matrix} \begin{bmatrix} 0.361 & 0.250 & 0.194 & 0.111 & 0.083 \\ 0.188 & 0.563 & 0.063 & 0 & 0.188 \\ 0.194 & 0.083 & 0.361 & 0.277 & 0.088 \\ 0.167 & 0 & 0.417 & 0.417 & 0 \\ 0.125 & 0.375 & 0.125 & 0 & 0.375 \end{bmatrix}$$

↑ the extend doc i is covered by doc 2

For d_1: $0.250 > 0.194$. cluster of d_2.

d_4: $0.417 > 0$ " " d_3.

d_5: $0.375 > 0.125$ " " d_2.

The clusters are $\{d_1, d_2, d_5\}$, $\{d_3, d_4\}$.

If we do col sums of matrix C, we get the total coverage of doc_i of col. i.

Calculation of c_{12}

$$c_{ij} = \alpha_i \sum_{k=1}^{n} d_{ik} \beta_k d_{jk}$$

$$c_{12} = \frac{1}{3}\left(d_{11}\beta_1 d_{21} + d_{12}\beta_2 d_{22} + \dots \right)$$

we are computing terms that will not contribute, a lot of times.

$d_1 \rightarrow t_1, t_4, t_6$

$d_2 \rightarrow t_1, t_2, t_3, t_4$

} d_{12} is 0.

Inverted Index for Seed Documents : IISD

seeds d_2, d_3

$t_1 \rightarrow d_2, d_3$

$t_2 \rightarrow d_2$

$t_3 \rightarrow d_2$

$t_4 \rightarrow d_2$

$t_5 \rightarrow d_3$

$t_6 \rightarrow d_3$

(For weighted matrices, we show $t_5 \rightarrow t_3$ as $t_5 \rightarrow \langle t_3, 1 \rangle$)

Cluster d_1. $d_1 \rightarrow t_1, t_4, t_6$.

- We need to calculate C_{12} and C_{13}.

Let $C_{12} = C_{13} = 0$.

(only update C_{12}, C_{13} for terms d_1 has)

For t_1 //

$$C_{12} = C_{12} + \alpha_1 \cdot (d_{11} \cdot B_1 \cdot d_{21})$$
$$= 0 + 1/3 \cdot (1 \cdot 1/4 \cdot 1) = 1/12$$

$$C_{13} = C_{13} + \alpha_1 \cdot (d_{11} \cdot B_1 \cdot d_{31})$$
$$= 0 + 1/3 \, (1 \cdot 1/4 \cdot 1) = 1/12$$

For t_4 //

$$C_{12} = C_{12} + \alpha_1 \cdot (d_{14} \cdot B_4 \cdot d_{24})$$
$$= 1/12 + 1/3 \cdot (1 \cdot 1/2 \cdot 1) = 0.250$$

For t_6 //

$$C_{13} = C_{13} + \alpha_1 \cdot (d_{16} \cdot B_6 \cdot d_{36})$$
$$= 1/12 + 1/3 \, (1 \cdot 1/3 \cdot 1) = 0.194$$

IISD:

1. Find seeds according to the seed power (and then acc. to # of terms shared w/ higher seeds)
2. Compute only C_{ij} $\forall$ non seed i and seed j.

   Start $C_{ij} = 0$.

   $C_{ij} = C_{ij} + \alpha_i \cdot d_{ik} \cdot \beta_k \cdot d_{jk}$ if $t_k \rightarrow d_i, d_j$

No IISD

$(m - n_c)(n_c)$

$\underbrace{(m - n_c)\, X_d}_{\text{non seed}} \cdot t_g$

IISD

$(m - n_c)\, X_d\, t_{g_s}$

t_{g_s}: avg length of inverted index for seeds

$t_{g_s} << t_g$

$O(m X_d t_{g_s})$

20.10.2023

cover coef.

all unique $n_c = m$ ⟷ all identical $n_c = 1$

$C_{m \times m}$ $C'_{n \times n}$

$n_c = n_c'$

$\Downarrow$

$1 < n_c \leq \min(m, n)$

for docs $\longrightarrow$ $\delta = \dfrac{\sum c_{ii}}{m} = \dfrac{\sum_{i=1}^{m} \delta_i}{m}$

$\delta = n_c/m \Rightarrow n_c = \delta . m$

for terms $\longrightarrow$ $\delta' = \dfrac{\sum_{i=1}^{n} c'_{ii}}{n} = \dfrac{\sum_{i=1}^{n} \delta'_i}{n} = \dfrac{n'_c}{n}$

$\Rightarrow n'_c = \delta' . n$

$n'_c = n_c \Rightarrow \delta' n = \delta m \Rightarrow \delta = \delta' \dfrac{n}{m}$

Observe $n_c = \dfrac{m.n}{t}$ ↳ ??? $D_1 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{2\times 3}$ $D_2 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}_{3\times 2}$

$t = 3$ $\dfrac{2.3}{3} = 2$ $\dfrac{3.2}{3} = 2$

Obtain $n_c = \dfrac{mn}{t}$ from $n_c = \sum_{i=1}^{m} \delta_i$:

$$c_{ij} = \alpha_i \sum_{k=1}^{n} d_{ik}\, \beta_k\, d_{jk}$$

$$\delta_i = c_{ii} = \alpha_i \sum_{j=1}^{n} d_{ij}^2\, \beta_j$$

$$n_c = \sum_{i=1}^{m} \alpha_i \sum_{j=1}^{n} d_{ij}^2\, \beta_j = \sum_{i=1}^{m} \sum_{j=1}^{n} \underline{\alpha_i}\, \underbrace{d_{ij}^2}_{\| \; d_{ij} \text{ (for binary matrices)}}\, \underline{\beta_j}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} \underbrace{\alpha_i}_{I} d_{ij} \underbrace{\beta_j}_{\searrow}$$

$$\left[\sum_{k=1}^{n} d_{ik}\right]^{-1} \quad \left[\sum_{k=1}^{m} d_{kj}\right]^{-1}$$

(column sum X_{d_i} — depth of indexing for d_i)

(raw sum tg_j — term generality for d_j)

$$\Rightarrow n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} \left[\sum_{k=1}^{n} d_{ik}\right]^{-1} \left[\sum_{k=1}^{m} d_{kj}\right]^{-1}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} \left[\underbrace{\sum_{k=1}^{n} d_{ik}}_{X_{d_i}} \cdot \underbrace{\sum_{k=1}^{m} d_{kj}}_{tg_j}\right]^{-1}$$

$$n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{d_{ij}}{X_{d_i}\, tg_j}$$

$$X_d = \sum_{i=1}^{m} \frac{X_{di}}{m} \qquad tg = \sum_{i=n}^{n} \frac{tg}{n}$$

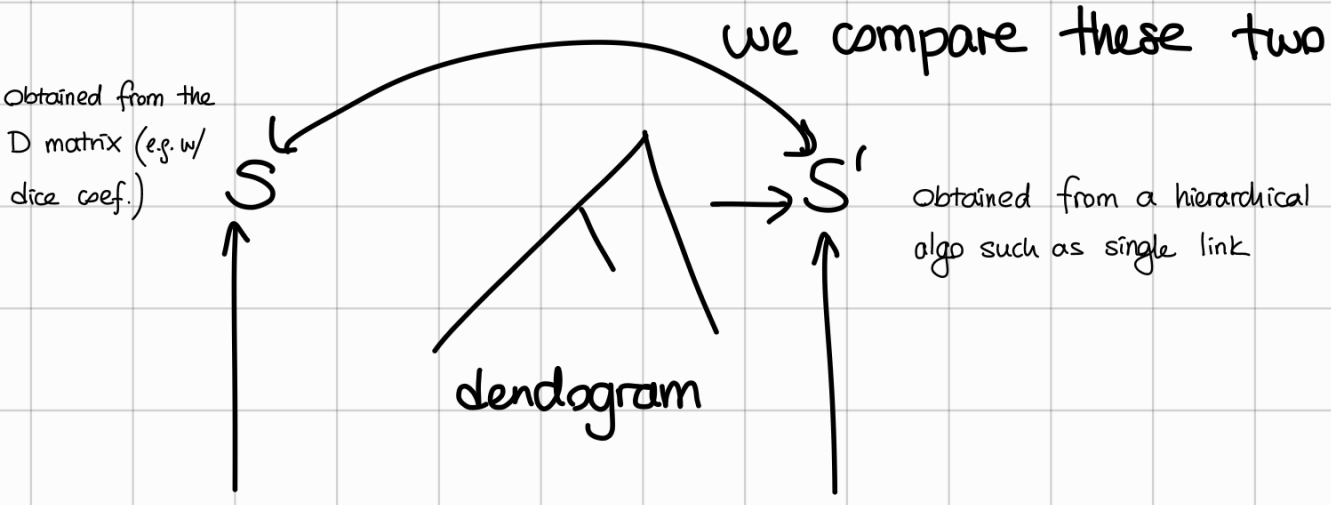
$$n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{d_{ij}}{X_d\, tg} = \frac{1}{X_d\, tg} \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}$$

$t =$ # of non zero entries in the matrix

$$x_d = \sum_{i=1}^{m} \frac{x_{di}}{m} = \frac{\sum_{i=1}^{m}\sum_{k=1}^{n} d_{ik}}{m} = \frac{t}{m}$$

Similarly for $t_g = \frac{t}{n} \Rightarrow n_c = \frac{t}{x_d t_g} = \frac{t}{\frac{t}{m}\cdot\frac{t}{n}} = \frac{mn}{t}$

$$n_c = \frac{mn}{t} \quad \square.$$

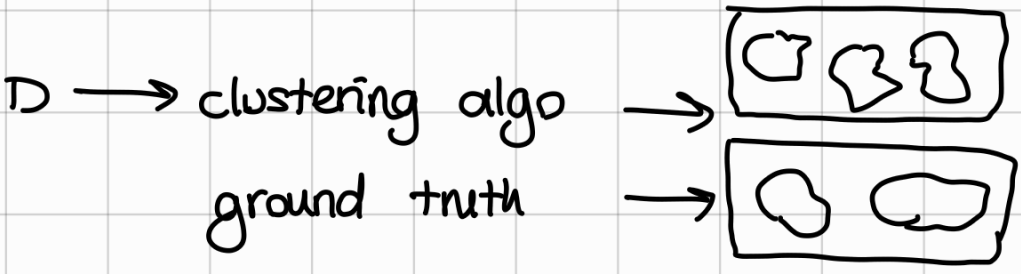


ab 0.6

bc 0.5

⋮

Rank similarity among pairs of docs



cluster purity = how homogenous clusters are

Rand Index

We have cluster output (partitions) (a b) (c d') (e' f')

" " Human based ground truth (or gold standard)

(a bc) (d' e' f')

	a	b	c	d'	e'	f'
a	-	ab	ac	ad'	ae'	af'
b	-	-	bc	bd'	be'	bf'
c						
d'						
e'					e'e'	
f'						

$$\text{Rand Index} = \frac{TP+TN}{TP+TN+FN+FP}$$

Pair	ab	ac	ad'	ae'	af'	bc	bd'	be'	bf'	cd'	ce'	cf'	de'	d'f'	e'f'
Class	TP	FN	TN	TN	TN	FN	TN	TN	TN	FP	TN	TN	FN	FN	TP

↑

TP : true pos.

TN

FP

FN

Corrected Rand Coef

- it takes out matches due to randomness

24.10.2023

Cluster Purity

- we need to have a gold standard
- we have a partitioning structure

IR:1 OS:3 DB:0

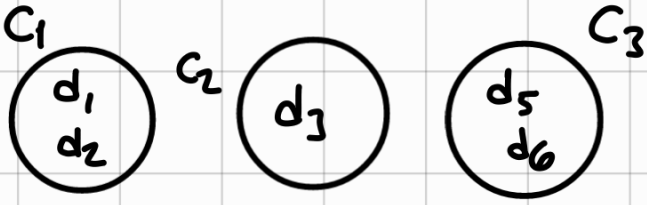
IR:3 OS:2 DB:1

IR:1 OS:2 DB:3

$$\text{cluster purity} \rightarrow \frac{3+3+3}{4+6+6} = \frac{9}{16} = \frac{\sum_i \max c_i}{\sum_i \sum_j (j \in c_i)}$$

$q_1 \rightarrow 1, 2, 3$

$q_2 \rightarrow 1, 5, 6$



n_t : # target cluster

$q_1 \rightarrow C_1, C_2 \quad n_{tr_1}$

$q_2 \rightarrow C_1, C_3 \quad n_{tr_2}$

$$n_t = \frac{2+2}{2} = 2$$

n_t ↑ avg # target cluster

Cluster Validation

1. Keep the clustering struct. the same
2. Distribute docs randomly to clusters
3. Find the avg. # target clusters for the random query? n_{tr}

For a meaningful structure we observe:

$n_t \ll n_{tr}$

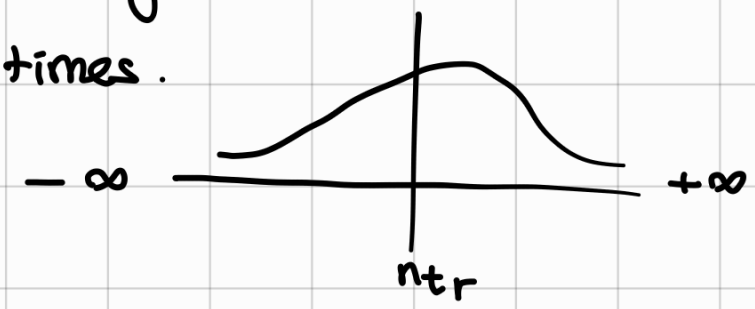
$n_t < n_{tr}$ → we have hope

we need to show $n_t \ll n_{tr}$

$n_t \geq n_{tr}$ ↑ no good don't proceed

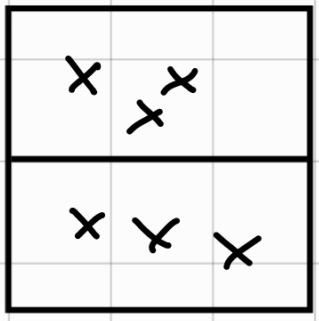
Is there a formula for n_{tr}?

(Without having to do monte Carlo simulation 1000s times.)



S. B. Yao, Com. of ACM 1970

Disk block records



Yao's Derivation : n = # records

m = # blocks

block size $= p = n/m$ ← # records in j^{th} block (since each block has same # records)

$n - p$ = # records in the other blocks

k = # records to be accessed

$$k < n - \frac{n}{m}$$

(n: all; $\frac{n}{m}$: except one block)

Remark.

$$C_k^n = \frac{n!}{k!\,(n-k)!}$$

(e.g. a, b, c $\quad C_2^3 = \frac{3!}{2!\,1!} = 3$ $\quad ab, ac, bc$)

C_k^{n-p} = different ways of selecting k records from $n-p$ records

The prob. of selecting no records from the i^{th} block:

$$\frac{C_k^{n-p}}{C_k^n}$$

$$n-p = n - \frac{n}{m} = n\underbrace{\left(1-\frac{1}{m}\right)}_{d} = nd$$

$E(I_j)$ = expected value of selecting at least a record from the i^{th} block

$$= \left(1 - \frac{C_k^{nd}}{C_k^n}\right)$$

Expected number of blocks to be accessed

$$= \sum_{j=1}^{m} E(I_j) = m\,E(I_j)$$

$$n_{tr} = \curvearrowleft = m \times \left(1 - \frac{C_k^{nd}}{C_k^n}\right) = m\left[1 - \frac{\frac{nd!}{k!\,(nd-k)!}}{\frac{n!}{k!\,(n-k)!}}\right]$$

$$= m \cdot \left[1 - \frac{\frac{nd!}{(nd-k)!}}{\frac{n!}{(n-k)!}} \right] = m \cdot \left[1 - \frac{(nd)!}{(nd-k)!} \cdot \frac{(n-k)!}{n!} \right]$$

$$= m \cdot \left[1 - \underbrace{\frac{1.2. \ldots . nd}{nd \cdot (nd-1) \ldots (nd-k)} \cdot \frac{1.2 \ldots (n-k)}{1.2 \ldots n}} \right]$$

$$\frac{(nd-k+1) \ldots (nd)}{(n-k+1) \ldots n}$$

$$= m \cdot \prod_{i=1}^{k} \frac{nd-i+1}{n-i+1} = n_{tr} \leftarrow \text{avg \# target clusters for random query}$$

How to apply this for clustering?

- Using our notation

  m = # docs k = # docs we want to access

  m_j = # docs other than cluster j we consider

let $m = 100$, $k = 3$. $|C_1| = 5$ $C_1 \rightarrow$ cluster-1

$m_1 = 100 - 5$

P_j = Prob. that we have a doc coming from C_1

Remark:

$$E(I_j) = 1 - \frac{C_k^{nd}}{C_k^n} = \prod_{i=1}^{k} \frac{nd - i + 1}{n - i + 1}$$

$$P_j = E(I_j) = \prod_{i=1}^{k} \frac{m_j - i + 1}{m - i + 1}$$

Expected # clusters to be accessed $= n_{tr}$

$$n_{tr} = \sum_{i=1}^{n_c} P_j \qquad (n_c : \text{\# clusters})$$

$$P_1 = \left[1 - \prod_{i=1}^{3} \frac{95 - i + 1}{100 - i + 1} \right]$$

27.10.2023

Inverted File Processing

Inverted Index for Search Engines, Zobel Moffat

Idea: skips & sorts
↳ (self indexing inverted files)

Boolean Retrieval

Conjunctive queries and (can be extended to or)

doc no ↑ → # occurences of t_1 in doc no

$t_1 \rightarrow \langle 1,1\rangle, \langle 5,1\rangle, \langle 20,1\rangle \langle 40,1\rangle$

$t_2 \rightarrow \langle 1,1\rangle, \langle 30,1\rangle, \langle 100,1\rangle$

posting lists

$Q = t_1, t_2$

Doc id.	# comparisons	
1	1	1:1
5	1	5:30 5<30
20	1	20:30 20<30
40	2	40:30, 40:100 40>30, 40>100

How to decrease the size of posting list?

⟨5,1⟩⟨8,1⟩⟨12,2⟩⟨13,1⟩⟨15,1⟩⟨18,1⟩⟨23,2⟩⟨28,1⟩⟨29,1⟩

1. Use d-gaps (document gaps):

use not the doc id, but the diff between the ids

⟨5,1⟩⟨3,1⟩⟨4,2⟩⟨1,1⟩⟨2,1⟩⟨3,1⟩⟨5,2⟩⟨5,1⟩⟨4,1⟩

2. Use skips every 3 posting list entry:

⟨5,1⟩⟨8,1⟩⟨12,2⟩⟨13,1⟩⟨15,1⟩⟨18,1⟩⟨23,2⟩⟨28,1⟩⟨29,1⟩

[13,] [23,] [-, 0]

e.g. Compare w/ [-,0] ⟨100,1⟩⟨105,5⟩

dgap – 9 comparisons
compare 100 w/ each
doc. 100 > 29 so don't
compare 105 at all

skips — 5 comparisons
compare 100 w/ 13
w/ 23
w/ -
→ so compare w/ 28,29 too.

3. Skips w/ d-gaps:

[3,] ⟨5,1⟩⟨3,1⟩⟨4,2⟩ [23,] ⟨1,1⟩⟨2,1⟩⟨3,1⟩ [-,0] ⟨5,2⟩⟨5,1⟩⟨4,1⟩

4. Frequency Ordered Inverted List (FOIL):

Reg. ⟨12,2⟩⟨17,2⟩⟨29,1⟩⟨32,1⟩⟨40,6⟩⟨78,1⟩⟨101,3⟩⟨106,1⟩

FOIL ⟨6:1:40⟩⟨3:1:101⟩⟨2:2:12,17⟩⟨1:4:29,32,78,106⟩

$f_{d,t}$: freq. of term t in the docs

docs w/ this freq.

doc list

5. Ordering posting lists acc. to retrieval numbers

Sengör, Altingövde METU

$t_1 \rightarrow \underbrace{[10]\langle d_2, 2\rangle}_{\uparrow}\langle d_5, 2\rangle [8]\langle d_3, 4\rangle [5]\langle d_7, 2\rangle\langle d_{10}, 8\rangle$

d_2 is accessed 10 times in the list of retrieved docs.

Ranked Retrieval

$f_{d,t}$: frequency of term t in doc d

$f_{q,t}$: " " " in query q

f_t : # docs containing term t

F_t : # term t in the collection

N : # docs

n : # terms

e.g. $D = \begin{matrix} & t_1 & t_2 & t_3 & t_4 \\ \end{matrix}$

$$\begin{bmatrix} 5 & 1 & 0 & 2 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 4 & 1 \end{bmatrix} \begin{matrix} d_1 \\ d_2 \\ d_3 \end{matrix}$$

$f_{d,t} \mid f_{1,1} = 5$

$f_t \mid f_1 = 3$

$F_t \mid F_1 = 5+2+1 = 8$

$N = 3$

$n = 4$

Assigning weights to doc & query terms

Assign less weight to terms w large f_t $w_{q,t} \propto \frac{1}{f_t}$
(occurs in many docs)

more weight is given to terms that occur frequently in a doc $\frac{F_t}{f_t}$, $f_{d,t} \propto w_{q,t}$

less weight to docs that contain many terms

?

tf.idf

Term weight assignment. Salton & Buckley

Q: $w_{q,t} = \ln\left(1 + \frac{N}{f_t}\right)$ weight of term t in q.

D: $w_{d,t} = 1 + \ln f_{d,t}$

$$w_q = \sqrt{\sum_t w_{q,t}^2}$$

$$w_d = \sqrt{\sum_t w_{d,t}^2}$$

$$S_{q,d} = \frac{\sum_t w_{d,t} \cdot w_{q,t}}{w_d \cdot w_q}$$

Salton & Buckley

- term freq. component
- doc. "
- normalization

Assume that the similarity of a doc to a query is the total no. of these terms in the docs.

$Q: w_1$ & w_2

d_1: $w_1 : 10$, $w_2 : 5$

d_2: $w_1 : 5$, $w_2 : 1$

$sim_{d_1} : 15$ $\quad$ $sim_{d_2} : 6$

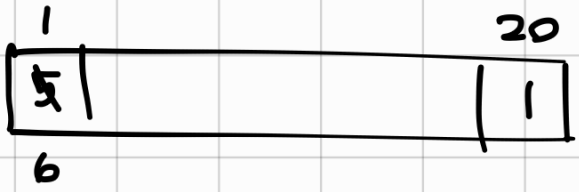
$q: w_1$ & w_2

$w_1 \rightarrow \langle d_1, 5\rangle\langle d_7, 2\rangle\langle d_{11}, 2\rangle\langle d_{20}, 1\rangle$

$w_2 \rightarrow \langle d_1, 1\rangle\langle d_4, 1\rangle\langle d_7, 10\rangle$

remark. w_i & t_i both mean term i.

Rank the entries



Assume that we have 3 accumulators

0	0	0

Process w_1 (w_1 is more important than w_2)

5	2	2
d_1	d_7	d_{11}

Process w_2

~~5~~	~~2~~	2
6	12	↑ no change

→ Ranking $d_1 : 6$ $d_7 : 12$ $d_{11} : 2$

Organize terms according to their freq. in a doc.

$w_1 \rightarrow \langle d_1, 5\rangle\langle d_7, 2\rangle\langle d_{11}, 2\rangle\langle d_{20}, 1\rangle$

$w_2 \rightarrow \langle d_7, 10\rangle\langle d_1, 1\rangle\langle d_4, 1\rangle$

0	0	0

d_7	d_1	d_4
10	1	1

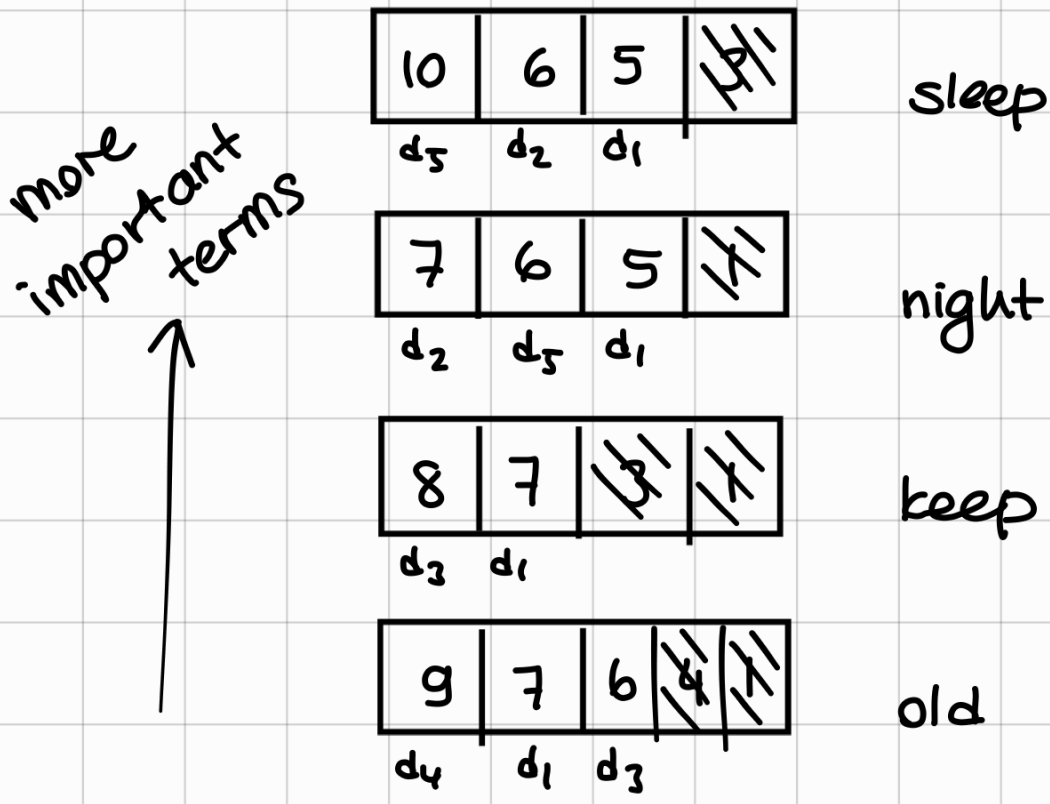
process w_2

12	6	
~~12~~	~~6~~	1

process w_1

$d_7 : 12, d_1 : 6, d_4 : 1$

Interleaved processing of query terms



Accumulators

31.10.2023

conjunctive query = and

disjunctive " = or

doc_1
$w_1 = 5$
$w_2 = 2$

doc
$w_1 = 2$
$w_2 = 1$

query w_1 and w_2 doc_1 = total weight 7

doc " " 3

$w_1 \rightarrow \langle d_1, 5 \rangle \langle d_7, 2 \rangle \langle d_{11}, 2 \rangle \langle d_{20}, 1 \rangle$

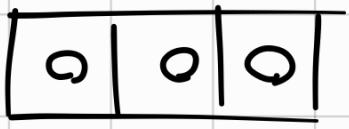
$w_2 \rightarrow \langle d_1, 1 \rangle \langle d_4, 1 \rangle \langle d_7, 10 \rangle$

Accumulators

6	0	0	1	. . .
d_1	d_2	d_3	d_4	

Limited # of Accumulators

-3 acc.s



5 2 2

6 12

d_7 : 12

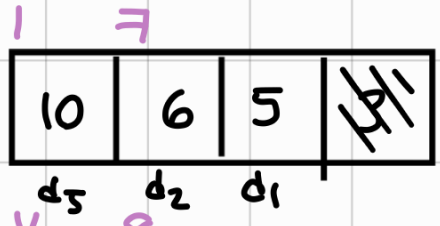
d_1 : 6

d_{11} : 2

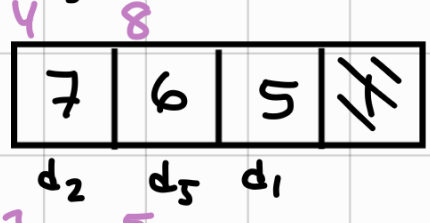
We left here last lecture

Interleaved processing of query terms

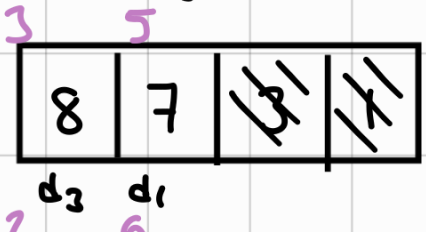
more important terms



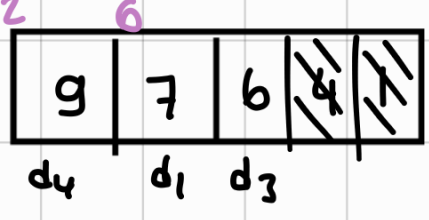
sleep



night



keep



old

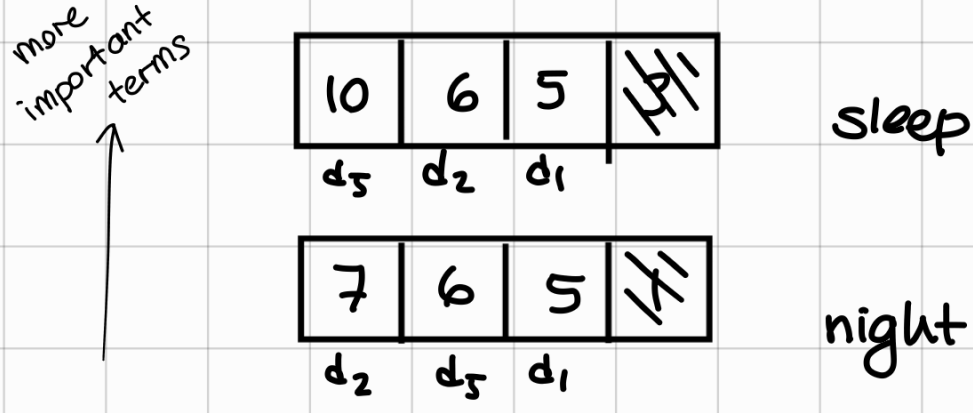
first we care about the freq. of the term than the importance of the term

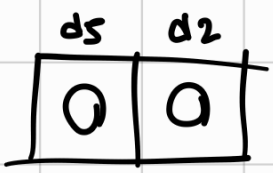
Accumulators

1	2	3	4	5	6	7	
0	0	0	0	0	0	0	. . .

sleep	10 / d_5				10
old	9 / d_4			9	10
keep	8 / d_3		8	9	10
night	7 / d_2	7	8	9	10
keep	7 / d_1	7	8	9	10
⋮	8 more lines				
		24	13	14	19	16

Interleaved processing of query terms w/ limited # accumulators





sleep	10 / d_5	10 0	Ranking
night	7 / d_2	10 7	d_5 16
sleep	6 / d_2	10 13	d_2 13
night	6 / d_5	16 13	

--- stop since we don't have space for another doc.

03.11.2023

Term Weighting

- Term weighting approaches in automatic text retrieval – Salton & Buckley 1988

Relational model of data 1970

No. of different weighting function (matching function)

MF

I think it is matching terms w/ docs or queries

Components of Matching Functions

TFC: term freq. component

CFC: collection freq. ⇓

NC: normalization ⇓ • doc • query (no NC needed)

TFC

b: binary 0,1

t: raw frequency

n: augmented normalized term frequency $= 0.5 + \frac{t_f}{\max(t_f)} \cdot 0.5$

We have 3 options to choose as Either b (binary TF), t (raw TF, as is) or n (augmented normalized TF)

applies to nonzero t_f's.

$t_f = 0 \Rightarrow n = 0$.

↳ term occurrence?

eg. $(5\ 0\ 1\ 2) \Rightarrow \max(t_f) = 5 = \max(\{5,0,1,2\})$

b: $(1\ 0\ 1\ 1)$

t: $(5\ 0\ 1\ 2)$

n: $\left(0.5 + \frac{5 \cdot 0.5}{5}\ \ 0\ \ \underbrace{0.5 + \frac{0.5}{5} \cdot 1}_{0.6}\ \ \underbrace{0.5 + \frac{2 \cdot 0.5}{5}}_{0.7}\right)$

CFC

x: no change (use original TFC)

f: inverse collection frequency component

$$\ln\left(\frac{n}{tg_j}\right) + 1$$

(similarly for nonzero tf s)

n: # docs

tg_j: term generality for term j

: # unique docs containing term j

p: probabilistic inverse collection frequency factor

For CFC, either use x (TFC), f ($^{\text{inv. coll.}}_{\text{FC}}$) or p ($^{\text{probabilistic}}_{\text{inv. coll. FC}}$)

NC

2 options

x: no change

c: use cosine normalization

$$\frac{1}{\sqrt{\sum w_i^2}}$$

Different ways of assigning weights

Doc	Query
TFC: 3	TFC: 3
CFC: 3	CFC: 3
NC: 2	NC: 1

$(3.3.2) \times (3.3.1) = 162$

no normalization for query

e.g.

$$D = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{bmatrix} 2 & 0 & 1 & 2 & 0 \\ 0 & 2 & 1 & 3 & 1 \\ 2 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{array}{c} \max tf \\ 2 \\ 3 \\ 2 \\ 1 \\ 2 \end{array} \qquad m = 5$$

$4 \quad 2 \quad 2 \quad 4 \quad 3 \rightarrow tg_j$

For documents use tf_c

For doc. 1:

TFC : $t \rightarrow (2\ 0\ 1\ 2\ 0)$

CFC : $f \rightarrow \ell u \frac{m}{tg_j} + 1$

$(2 \quad 0 \quad 1 \quad 2 \quad 0)$

$\ell u \frac{5}{4}+1 \quad 0 \quad \ell u \frac{5}{2}+1 \quad \ell u \frac{5}{4}+1 \quad 0$

$\downarrow \quad \quad \downarrow \quad \quad \downarrow$

$1.22 \quad \quad 1.92 \quad \quad 1.22$

f	1.22	0	1.92	1.22	0
t	×2	×0	×1	×2	×0
tf	2.44	0	1.92	2.44	0

$$c = \frac{1}{\sqrt{\sum w_i^2}} = \frac{1}{\sqrt{2.44^2 + 1.92^2 + 2.44^2}} \cong \frac{1}{3.95}$$

$\hookrightarrow w_i = tf$

e.g. continued

t_{f_c}	$\frac{2.44}{3.95}$	0	$\frac{1.92}{3.95}$	$\frac{2.44}{3.95}$	0
doc1, but weighted ←	0.62	0	0.49	0.62	0

(for query normalization nfc doesn't make sense because it will not change rankings)

→ For Q use nfx

$Q = (1\ 0\ 0\ 2\ 0)$

		1	0	0	2	0
TFC	n	$0.5+\frac{1.05}{2}$	0	0	$0.5+\frac{2}{2}.05$	0
CFC	f: $\ell_h \frac{m}{tg_j} + 1$	1.22	1.92	1.92	1.22	1.51
no normalization → x	use nf as is	0.75.1.22 = 0.92	0	0	1.1.22	0

nfx: use n from TFC, f from CFC, and nothing from NC.

$Q = (0.92\ 0\ 0\ 1.22\ 0)$

$sim(Q,D) = \sum w_{qh} \cdot w_{dh}$

$$D = \begin{bmatrix} 0.62 & 0.00 & 0.49 & 0.62 & 0.00 \\ 0.00 & 0.66 & 0.33 & 0.63 & 0.26 \\ 0.78 & 0.00 & 0.00 & 0.39 & 0.48 \\ 0.63 & 0.00 & 0.00 & 0.00 & 0.78 \\ 0.73 & 0.58 & 0.00 & 0.37 & 0.00 \end{bmatrix}$$

$s(q, d_1) = 0.92 \cdot 0.62 + 0 + 0 + 1.22 \cdot 0.62 + 0 = 1.33$

$d_1 = 1.33$ $\quad d_4 = 0.58$

$d_3 = 1.29$

$d_5 = 1.12$

$d_2 = 0.77$

(remark. no normalization for queries since it won't change the rankings)

7.11.2023

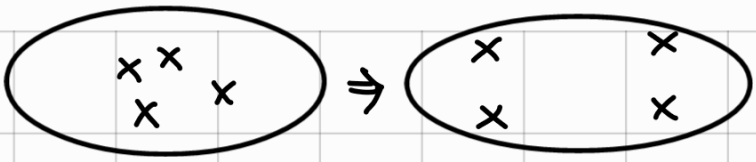
Term Discrimination Value

TDV: smth like importance of terms

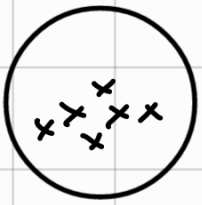
Learning TDV, some French team, SIGIR 2020.

- Find terms w/ TDV = 0.
- Delete them from indexing vocab.

⇒ IR becomes more efficient & more effective

- 

The use of a good discriminator will separate docs from each other.

-  The use of a bad discriminator will make docs less distinguishable.

How to calculate TDVs:

- similarity based
- # clusters

Space density: Q: Avg. similarity val. among docs

$$S = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}_{m \times n}$$

↳ symmetrical

$$Q = \frac{1}{\frac{m(m-1)}{2}} \cdot \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} S_{ij}$$

Q_j = avg. similarity without t_j.

$$Q \propto \text{similarity} \propto \frac{1}{\text{discrimination}} \propto \frac{1}{TDV}$$

Term Type	$Q : Q_j$	TDV
Good	$Q_j > Q$	> 0
Bad	$Q_j < Q$	< 0
Indifferent	$Q \approx Q_j$	≈ 0

→ If we add t_j Q decreases ⇓ t_j is a good discriminator ⇓ TDV of t_j is high

$$TDV = Q_j - Q$$

or

$$TDV = \frac{Q_j}{Q}$$

term weight: $TDV_\ell \cdot t_f$ (↳ TDV of term ℓ; ↳ term frequency)

ASK: of which term? t_{f_ℓ}?

How to calculate Q?

1. Obtain S matrix: expensive.
2. Obtain the collection centroid and calculate sim. of docs to the centroid.

$$D = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\text{Centroid} = \left(\frac{2}{3} \quad \frac{1}{3} \quad \frac{2}{3} \quad \frac{2}{3}\right)$$

Salton, ... Wang - Comm. ACM.

$$Q = \frac{1}{m} \sum_{i=1}^{m} Sim(d_i, centroid)$$ ← Vector Space Model

TDV Calculation using Clusters (Cover Coefficient Concept)

n_c = # clusters using all terms

n_{c_e} = # clusters w/o using term t_e

- A good discriminator should increase # clusters

Term Type	$n_c : n_{c_e}$	TDV
Good	$n_c > n_{c_e}$	$n_c - n_{c_e} > 0$
Bad	$n_c < n_{c_e}$	$n_c - n_{c_e} < 0$
Neutral	$n_c \approx n_{c_e}$	$n_c - n_{c_e} \approx 0$.

$$n_c = \sum_{i=1}^{m} C_{ii} = \sum_{i=1}^{n} \alpha_i \cdot (d_{i1}^2 \beta_1 + d_{i2}^2 \beta_2 + \ldots + d_{in}^2 \beta_n)$$

↳ cont.

JASIST 1987 paper

$$TDV_\ell = \sum_{i=1}^{f_\ell} \left[\delta_i - \alpha_i^{\ell} \left(\frac{\delta_i}{\alpha_i} - d_{i\ell}^2 \cdot \beta_\ell \right) \right]$$

$$\alpha'_{i\ell} = \left(\alpha_i^{-1} - d_{i\ell} \right)^{-1}$$

α_i^{-1}: original raw sum; $d_{i\ell}$: subtract the term ℓ

$$f_\ell = \{ d_i \mid d_i \in D \wedge d_{i\ell} \neq 0 \}$$

* More efficient calc of TDVs using cover coef. concept:

$$n_c = \frac{mn}{t} \qquad n_{c_\ell} = \frac{m(n-1)}{\left(t - \underbrace{\#\text{ docs that contain } t_\ell}_{\text{term generality of term } \ell \Rightarrow tg_\ell}\right)} = \frac{m(n-1)}{(t - tg_\ell)}$$

10.11.23

idf: inverse document frequency

TDV

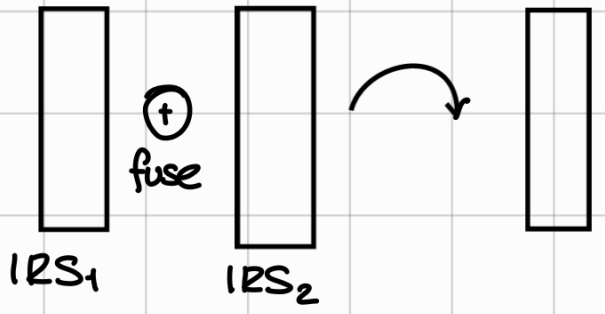
↳ idf ASK What do we mean here?

- term weighting

SIGIR 2020, Learning Term Disc. Values

Data Fusion

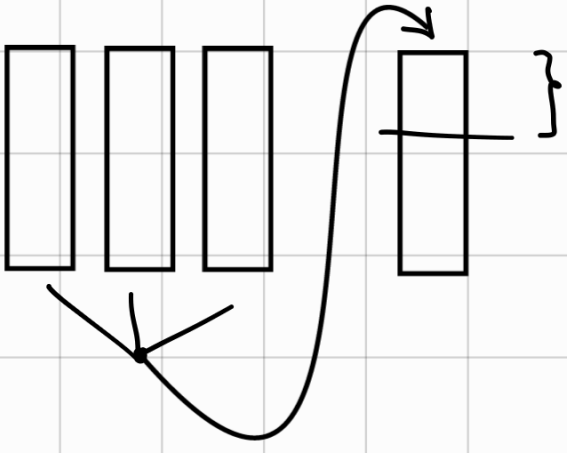
- Rank docs in diff. matching functions. Combine rankings
  ⇒ more effective IRS.



new ranking obtained by fusing the two rankings

- Meta search engines use data fusion

Data Fusion for Automatic Evaluation of Search Engines, IPM 2006, Nuray and Can



← Assume that they are relevant (pseudo relevant docs)

- Use pseudo relevant docs to rank IRSs according to their effectiveness.

Fusion Methods

1. Reciprocal rank
2. Borda Count
3. Condorcet

Bias Concept: being far away from the norm

↑ What do we use this for? For choosing which IRS to fuse.

1. Reciprocal Rank

$$r(d_i) = \frac{1}{\sum_{j=1} \frac{1}{\text{position } d_{ij}}}$$

↑ IRS

After finding all r values, rank acc. to r values (in the increasing order)

(i.e. lower r(i) = better rank)

Example.

IRS: A, B, C, D Documents: a, b, c, d, e, f, g

A: (a, b, c, d)

B: (a, d, b, e)

C: (c, a, f, e)

D: (b, g, e, f)

$$r(a) = \frac{1}{\frac{1}{1}+\frac{1}{1}+\frac{1}{2}} = \frac{1}{2.5} = 0.4$$

$$r(b) = \frac{1}{\frac{1}{2}+\frac{1}{3}+\frac{1}{1}} = \frac{6}{3+2+6} = \frac{6}{11} \approx 0.54$$

$$r(c) = \frac{1}{\frac{1}{3}+\frac{1}{1}} = \frac{3}{1+3} = 0.75$$

$$r(d) = \frac{1}{\frac{1}{4}+\frac{1}{2}} = \frac{4}{1+2} \approx 1.33$$

$$r(e) = \frac{1}{\frac{1}{4}+\frac{1}{4}+\frac{1}{3}} = \frac{12}{3+3+4} = \frac{12}{10} = 1.2$$

$$r(f) = \frac{1}{\frac{1}{3}+\frac{1}{4}} = \frac{12}{4+3} = \frac{12}{7} \approx 1.71$$

$$r(g) = \frac{1}{\frac{1}{2}} = 2$$

- Sort in increasing order:

$$0.4 < 0.54 < 0.75 < 1.2 < 1.33 < 1.71 < 2$$

a b c e d f g

Ranks: $a > b > c > e > d > f > g$.

2. Borda Count (Jean Charles, Cavalier de Borda)

- The highest ranked individual (in an n-votes) gets n votes, and each subsequent indv. gets one vote less; e.g. no 2 gets n-1 votes.
- If there are candidates left unranked by the voter, the remaining points are divided evenly among the unranked candidates.
- The choice with the highest number of votes wins

Example. $A = (a, c, b, d)$ $C = (c, a, b, e)$

$B = (b, c, a, e)$ Docs/candidates $= \{a, b, c, d, e\}$

$$BC(i) = BC_A(i) + BC_B(i) + BC_C(i)$$

$\#\,docs = n = 5. \Rightarrow$ max vote

$BC(a) = 5 + 3 + 4 = 12$

$BC(b) = 3 + 5 + 3 = 11$

$BC(c) = 4 + 4 + 5 = 13$

$BC(d) = 2 + 0 + 0 = 2$

$BC(e) = 0 + 2 + 2 = 4$

rankings:

$c > a > b > e > d.$

3. Condorcet Method (Marie Jean Antoine Nicholas de Caritat Condorcet)

The voting process takes "each preference of each voter for one candidate over another" into account.

Example. candidates / doc.s = a, b, c

Five voters = A, B, C, D, E

A : a > b > c

B : a > c > b

C : a > b = c

D : b > a

E : c > a

↳ Here, a's ranking is better than b, since b doesn't exist in the comparison

Each non diagonal entry (i, j) of the matrix below shows # votes i over j. (e.g. cell [a, b] shows # wins, # loses and # ties of doc. a over doc b)

(win, lose, tie)

	a	b	c
a	-	(4, 1, 0)	(2, 1, 0)
b	(1, 4, 0)	-	(2, 2, 0)
c	(1, 2, 0)	(2, 2, 0)	-

- At the end, document having the most wins get the first rank.
- If two docs have the same # wins, then the one w/ smaller # loses wins.
  else, tied.

won over b, c

	Win	Lose	Tie
a	2	0	0
b	0	1	1
c	0	1	1

tie with c

lost to a

ranks: $a > b = c$.

- We don't want to fuse all of the data coming from each IRS. We need to choose systems for data fusion.
- But how to choose?
  + Select systems far away from the avg behavior.

Choosing systems to fuse

Docs : a, b, c, d, e, f, g

$$A=\begin{bmatrix} a\ b\ c\ d \\ b\ a\ c\ d \\ a\ b\ c\ e \end{bmatrix} \begin{matrix} \rightarrow \text{results of query } Q_i \\ \rightarrow \quad " \quad " \quad " \quad Q_j \\ \rightarrow \quad " \quad " \quad " \quad Q_k \end{matrix} \begin{matrix} \leftarrow \\ \leftarrow \\ \leftarrow \end{matrix} \begin{bmatrix} b\ f\ c\ e \\ b\ c\ f\ g \\ c\ f\ g\ e \end{bmatrix} = B$$

a is retrieved 3 times

$$X_A = (3,3,3,2,1,0,0)$$

$$X_B = (0,2,3,0,2,3,2)$$

a b c d e f g

$$X = (3,5,6,2,3,3,2) \leftarrow \text{the norm}$$

$$S(V,W) = \frac{\sum V_i * W_i}{\sqrt{\sum v_i^2 * \sum w_i^2}}$$

$$Sim(X_A, X) = \frac{9+15+18+4+3+0+0}{\sqrt{(9+9+9+4+1)\cdot(9+25+36+4+9+9+2)}}$$

$$= 0.8841$$

$$Sim(X_B, X) = 0.8758$$

$$Bias(A) = 1 - Sim(X_A, X) = 0.1159$$

$$Bias(B) = 1 - Sim(X_B, X) = 0.1242$$

↳ B has higher bias

higher bias ⇒ being farther away from the norm.

Therefore we choose B during ranking.

14.11.2023

Maximal Margin Relevance (MMR)

- IR Query result diversification.

Query: Jaguar → Cat
→ Drink
→ Car
→ OS
} Which one?

ACM SIGIR, 1998

$$MMR = \underset{d_i \in R/S}{\arg\max} \left[\lambda Sim_1(d_i, Q) - (1-\lambda) \max_{d_j \in S} Sim_2(d_i, d_j) \right]$$

R: relevant doc.s in collection C

S: current result set

λ: hyperparameter

- High $\lambda \nearrow$ higher accuracy
- Low $\lambda \searrow$ " diversity

- The motivation create a set of docs that not only contains highly relevant info but also provides a diverse & comprehensive coverage of the topic.

- Showing multiple similar doc.s, even if they are relevant, introduces redundancy to the query results. MMR provides a balance between relevance & redundancy.

Example.

$S =$

	d_1	d_2	d_3	d_4	d_5	q (the query)
d_1	1	0.11	0.23	0.76	0.25	0.91
d_2		1	0.29	0.57	0.51	0.90
d_3			1	0.02	0.20	0.50
d_4				1	0.33	0.06
d_5					1	0.63
q						1

1st iteration: $S = \emptyset$, $R/S = \{d_1, d_2, d_3, d_4, d_5\}$

$$MMR = \underset{d_i \in R/S}{\arg\max} \left[\lambda \, Sim_1(d_i, Q) - (1-\lambda) \max_{d_j \in S} Sim_2(d_i, d_j) \right]$$

↳ $S = \emptyset$

$$MMR = \underset{d_i \in R/S}{\arg\max} \left(\lambda \, Sim_1(d_i, q) \right)$$

↳ doesn't change argmax here

$= d_1 \rightarrow$ has max sim w/ q.

$S = \{d_1\}$, $R/S = \{d_2, d_3, d_4, d_5\}$.

2nd iteration: $\lambda = 0.5$

$$MMR = \underset{d_i \in R/S}{\arg\max} \left[0.5 \left(Sim_1(d_i, Q) - \max_{d_j \in S} Sim_2(d_i, d_j) \right) \right]$$

$MMR(d_i)$

↳ λ doesn't affect results again.

$$MMR(d_i) = 0.5 \left[Sim_1(d_i, Q) - Sim_2(d_i, d_1) \right]$$

$MMR(d_2) = 0.5(0.90 - 0.11) = 0.395$

$MMR(d_3) = 0.5(0.50 - 0.23) = 0.135$

$$MMR(d_4) = 0.5(0.06 - 0.76) = -0.35$$

$$MMR(d_5) = 0.5(0.63 - 0.25) = 0.19$$

$$MMR = \underset{d_i \in R/S}{\arg\max} \left(MMR(d_i) \right) = d_2.$$

Update:

$$S = \{d_1, d_2\} \qquad R/S = \{d_3, d_4, d_5\}$$

3rd iteration:

$$MMR = \underset{d_i \in R/S}{\arg\max} \left[\overbrace{0.5\left(Sim_1(d_i, Q) - \max_{d_j \in S} Sim_2(d_i, d_j) \right)}^{MMR(d_i)} \right]$$

$$MMR(d_3) = 0.5\left(0.50 - \max\{Sim_2(d_3, d_2), Sim_2(d_3, d_1)\}\right)$$
$$= 0.5\left(0.50 - \max\{0.23, 0.29\}\right)$$
$$= 0.50\left(0.50 - 0.29\right) = 0.105$$

$$MMR(d_4) = 0.5\left(0.06 - \max\{0.76, 0.57\}\right) = -0.35$$

$$MMR(d_5) = 0.5\left(0.63 - \max\{0.25, 0.51\}\right) = 0.065$$

$$MMR = \underset{d_i \in R/S}{\operatorname{argmax}}\ MMR(d_i) = d_3$$

Update :

$$S = \{d_1, d_2, d_3\} \qquad R/S = \{d_4, d_5\}$$

After 3rd iteration for $\lambda = 5$, S = result set = $\{d_1, d_2, d_3\}$.

Total pairwise similarity for S:

$$sim(d_1, d_2) + sim(d_2, d_3) + sim(d_1, d_3) = 0.63$$

For $\lambda = 1$, S' would be $\{d_1, d_2, d_5\}$.

Total pairwise similarity for S':

$$sim(d_1, d_2) + sim(d_2, d_5) + sim(d_1, d_5) = 0.97.$$

$\lambda \downarrow$, diversity $\uparrow$

21.11.2023

PAT trees & PAT Arrays

(used for string search)

Info ret.: algos & data structures

Example. binary alphabet (semi infinite string)

1 2 3 4 5 6 7 8 9 10 11 12

0 1 1 0 0 1 0 0 0 1 0 1

↳ sis

sis1: 0 1 1 0 0 1 0 0 0 1 0 1 ...

sis2: 1 1 0 0 1 0 0 0 1 0 1 ...

sis3: 1 0 0 1 0 0 0 1 0 1 ...

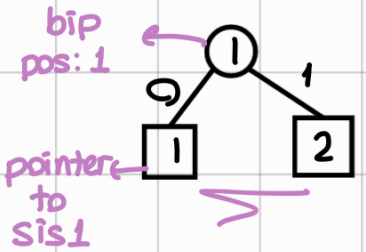
sis4: 0 0 1 0 0 0 1 0 1 ...

sis5: 0 1 0 0 0 1 0 1 ...

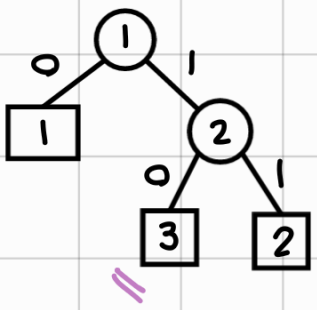
sis6: 1 0 0 0 1 0 1 ...

sis7: 0 0 0 1 0 1 ...

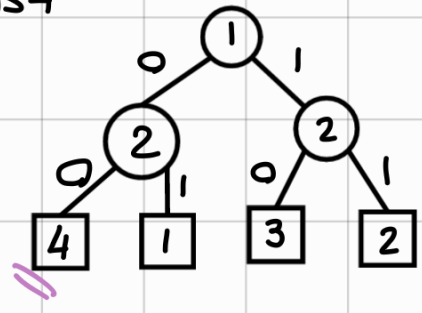
sis1 & sis2



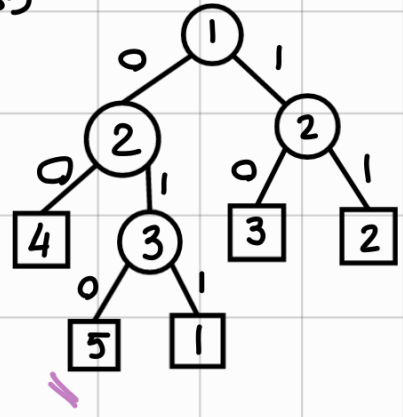
sis3



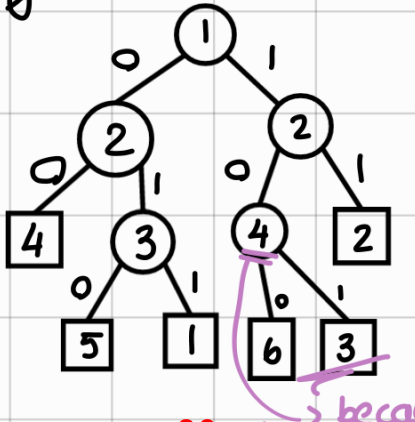
sis4



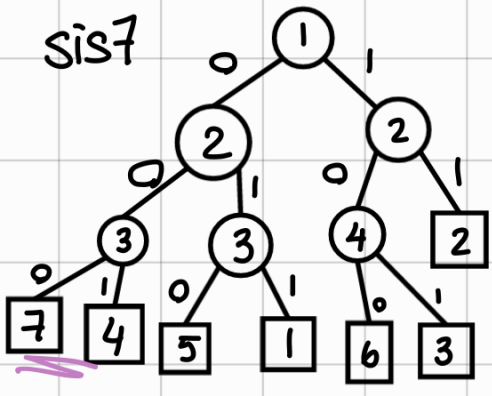
sis5



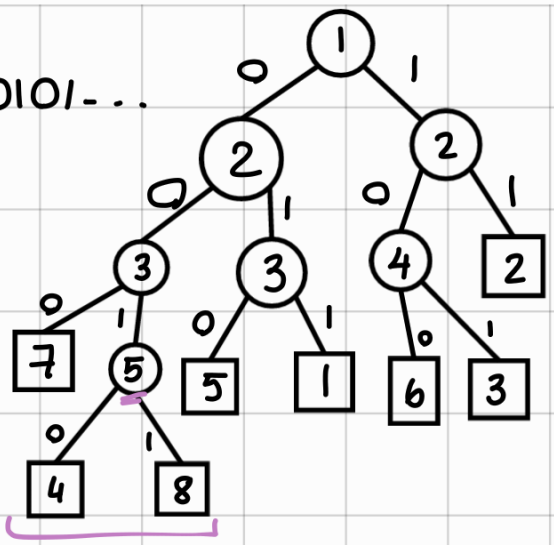
sis6



sis7



S[1:8]: 00101...



PAT Array

011 1001

(bit patterns) 7, 4, 8, 5, 1, 6, 3, 2 — Gonnet et. al

0001 00100 01100 100 1100

00101

→ increasing order

- Perform a binary search in the PAT array to see if a string appears for a string query.

  (You can keep the PAT tree as well,

  ⚠ but you don't need to keep the tree structure, the array is enough for binary search.)

  - Look at the middle item, if query > item : search right subtree ; else, left subtree.

- This method is for binary alphabet.

PAT tree of a text:

- same principle, we should be able to decide $>, <, =$

sis1: that is that that is not that

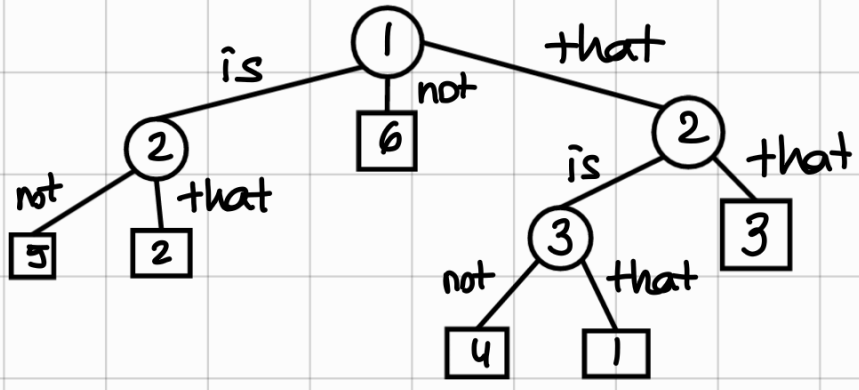
sis2: is that that is not that

sis3: that that is not that

sis4: that is not that

sis5: is not that

sis6: not that



Proximity Search

string1 $\overset{\text{dist}}{\longleftrightarrow}$ string2

A string s1 is at most a fixed # characters away from another string s2.

Distance between 01 and 10

5,1 — 6,3

what does this mean?

ASK

28.11.2023

Evaluation Under Diversity

Query: James Bond

Actors, Music, Director, Places, Singer

→ Ambiguous query

Remark: we used MMR w/ low λ to increase diversity of query results.

$$\text{S-Recall (Subtopic recall)} = \frac{\left(\begin{array}{c}\text{Unique \# topics covered} \\ \text{until } n^{th} \text{ rank}\end{array}\right)}{(\text{Total \# topics})}$$

Example.

Rank	Document	Subtopic
1	d_1	m_3
2	d_2	m_4
3	d_3	m_1, m_2
4	d_4	m_5, m_6
5	d_5	m_6
6	d_6	m_5
7	d_7	m_4
8	d_8	m_3
9	d_9	m_2
10	d_{10}	m_1

Assume $\exists$ 6 subtopics

$$\text{S-Recall@2} = \frac{|\{m_3, m_4\}|}{6} = 2/6 \approx 0.33$$

$$\text{S-Recall@6} = \frac{|\{m_1, m_2 \ldots m_6\}|}{6} = 6/6 = 1.$$

Rank	m_1	m_2	m_3	m_4	m_5	m_6
1			1			
2				1		
3	1	1				
4					1	1
5						1
6					1	
7				1		
8			1			
9		1				
10	1					
P@5	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{2}{5}$

IA - Precision

Precision - IA

Intent Aware

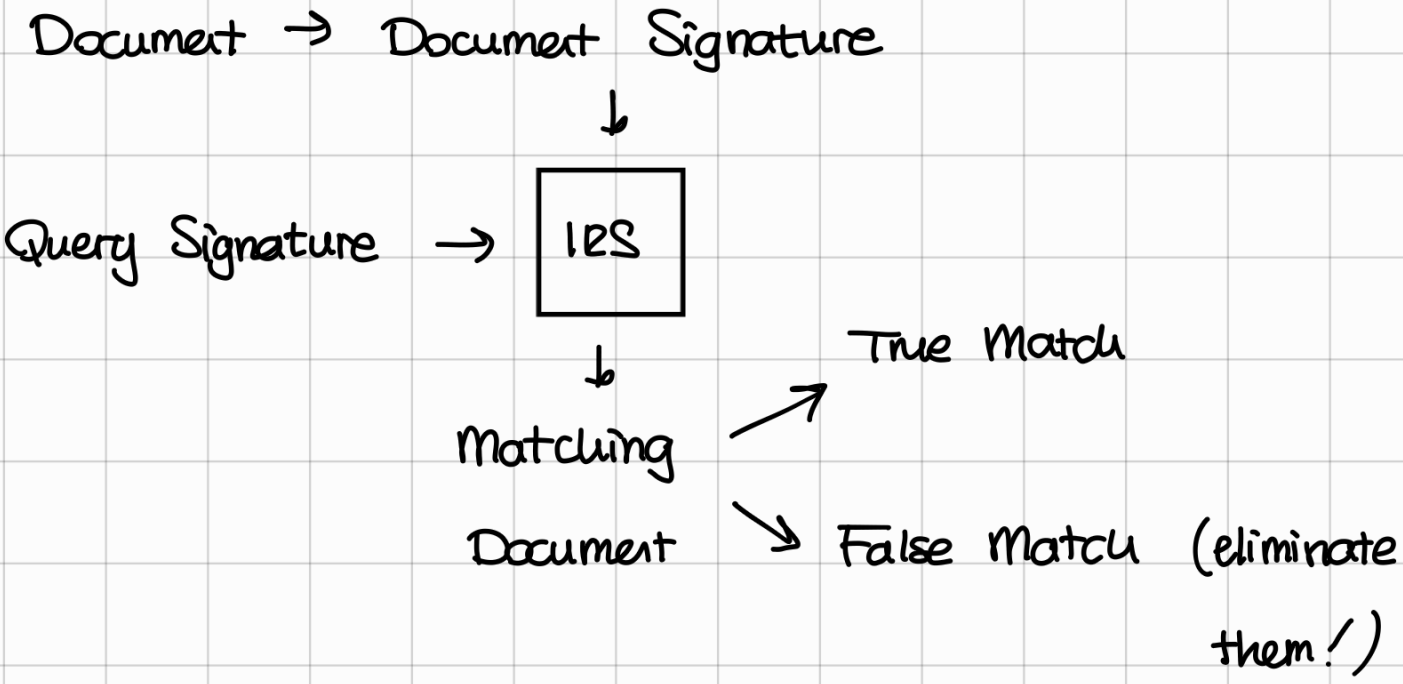
$$\text{Precision-IA @5} = \frac{1}{|m_i|} \sum_{m_i} P@5$$

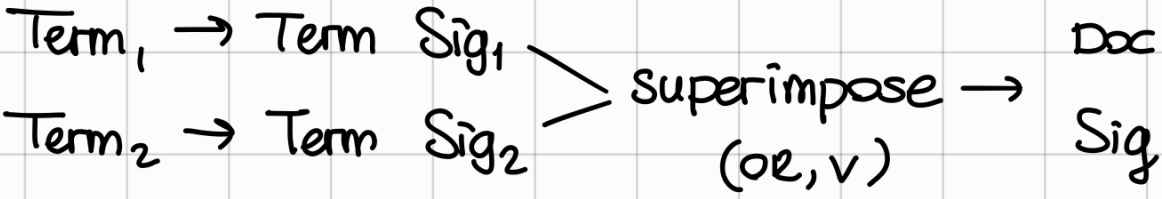
$$= \frac{1}{6} \cdot \left(\frac{1}{5} + \frac{1}{5} + \frac{1}{5} + \frac{1}{5} + \frac{1}{5} + \frac{2}{5} \right)$$

$$\approx 0.23$$

- Recommendations should cover diff. types of meanings
- NOTE: this topic "eval. under div." belongs w/ MMR.

Signature Files





F = signature size = # bits ≈ 500 bits

- Now let's assume sig. size F = 8.

Term	Term Signature	
Object	1000	1000
Signature	0010	0100
Generation	1000	0100
	1010	1100 ← doc sig. (DS)

m = # bits to be set $\forall$ term i.

(for each term)

Object → obtain its doc equivalent

Consider conjunctive queries = q_1 & q_2

Query → Generation

Query Sig. (QS) : 1000 0100

⚠ if Q_S & D_S == Q_S , then retrieve doc.

else , skip.

Example.

D_S = 1010 1100

Q_S = 1000 0100

1000 0100 = Q_S → doc contains the term, true match

- For matchers, obtain the doc. Make sure it contains the term.

Query : "Operation" Q_s = 1000 0100 (why is it the same as query "generation"?)

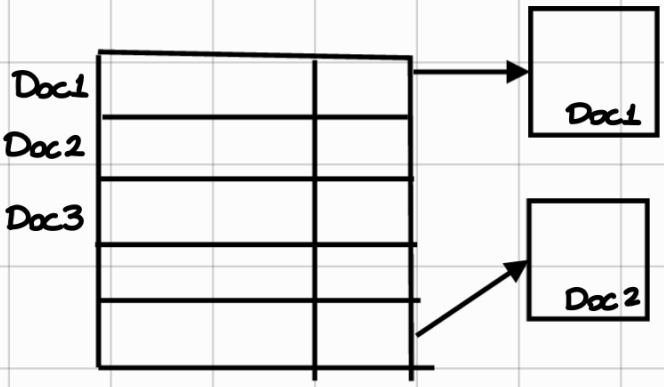
D_s = 1010 1100

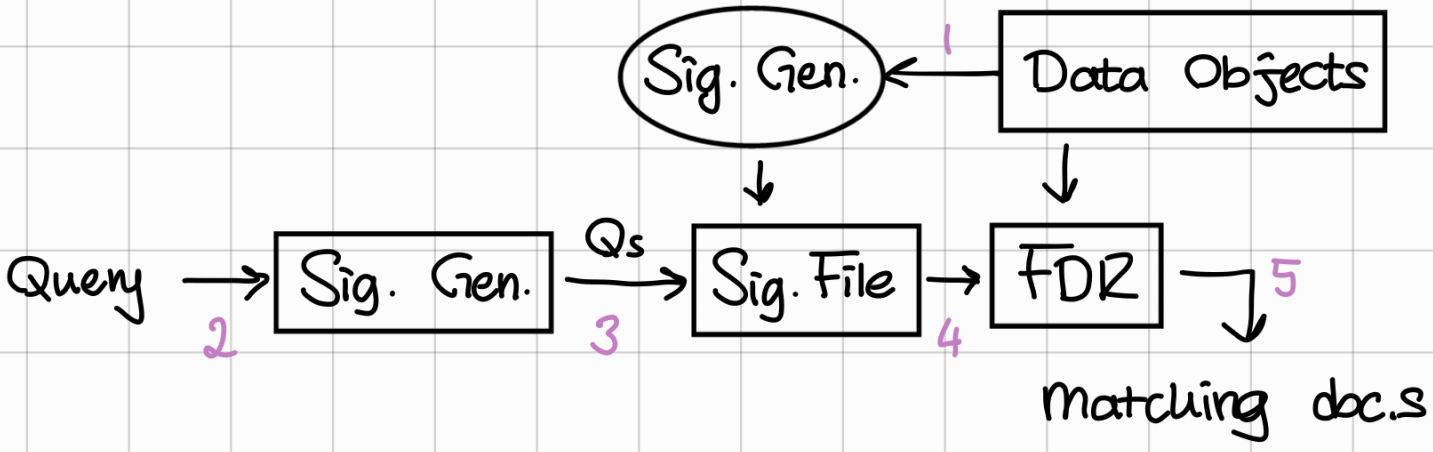
Q_s = 1000 0100

1000 0100 = Q_s

This is false match because "operation" term is not in our real doc.

False Drop Elimination = False Drop Resolution (FDR)





- MIT MS 1991 (Christodolacy?, Faloutsos, Astor Zobel)
  UNN of Toronto

- Doc.s are divided into blocks w/ the same # unique term.

F = signature size (fixed size)

D = avg # unique term / block

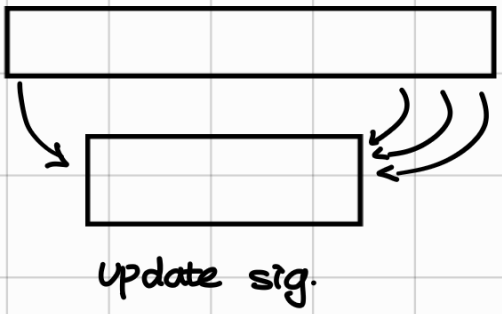
m = # bits to set per term

$$m = \frac{F \ln 2}{D}$$

→ Prob. of having a '0' or a '1' is the same ∀ bits positions of

- Lee & Leng 1993

They aimed to generate block signature w/ the same # '1's.



- In a doc. sig.

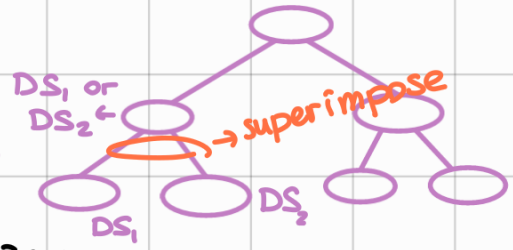
  # "1"s = # "0"

- We will see different signature file organizations:
  - extendible hashing
  - linear hashing

Storage Structures for Signature Files

- Aims to avoid query – doc signature file comparison.

1. Single Level
   - Sequential (problem here is we would have to compare all)
2. Multi Level
   - signature tree →



3. Vertical Partitioning
   - bit sliced
   - frame sliced
4. Horizontal Partitioning
   - fixed prefix
   - extendible hashing
   - linear hashing

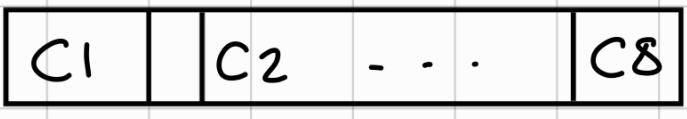
01.12.2023

Bit Sliced Signature Files

S_1: 0001 1110
S_2: 1101 0001
S_3: 0011 1100
S_4: 1100 0011
S_5: 0011 0110
S_6: 1100 1001

columns of these sigs →

C1: 0 1 0 1 0 1
C2: 0 1 0 1 0 1
C3: 0 0 1 0 1 0
C4: 1 1 1 0 1 0
C5: 1 0 1 0 0 1
C6: 1 0 1 0 1 0
C7: 1 0 0 1 0 0
C8: 0 1 0 1 0 1



↑ extra new space for possible new columns?

- Insertion of a new signature is difficult, since col.s are stored one after another. That's why we store an extra space for possible new comes.

Ex. Q_s = 1011 0000

↓ column1 → column3

C1 & C3

0	0	0	None of the 6 sig.s have the terms in Q_s. Don't proceed further.
1	0	0	
0 ∧	1 =	0	
1	0	0	
0	1	0	
1	0	0	

↳ intermediate result

We compared 6 bits insted of 8. ☺

(works when $\frac{|sigs|}{6} < \frac{|columns\ in\ a\ sig|}{8}$)

If not all sigs would have 0, we would proceed with column 4, by (intermediate result) ∧ C4.

- Here bit density is 50% (half "0"s, half "1"s)
  ⇒ ∀ step (of this col. comparison), we eliminate half of them. → ASK: half of col.s or sig.s?

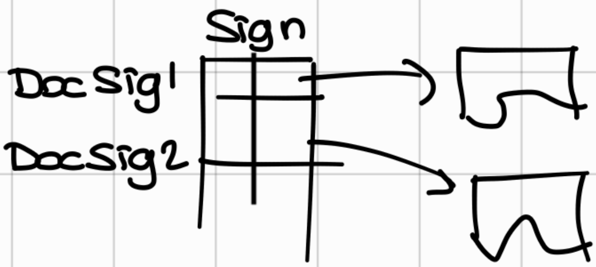
- If remaining "1"s are so small, we switch to FDR. Since now FDR is less expensive than ANDing.

– Karaberber & Can: Partial Eval. of Queries for Bit-Sliced Sig. Files

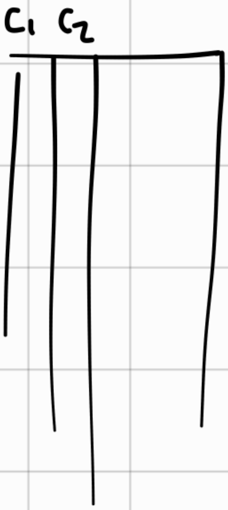
05.12.2023

Match → False Match / True Match } for signature matching

Falsedrop resolution: elimination of false drops. (false matches)



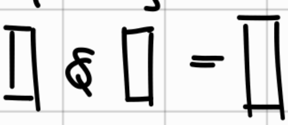
(this is serial (1 processor) but not sequential (compare all docs to the query))



← Bit sliced signature processing

$Q_5 = 1011\ 0000$

c_1 c_3



50% "0"

50% "1"

Example. (we are repeating from a prev. lecture)

S_1: 0001 1110

S_2: 1101 0001

S_3: 0011 1100

S_4: 1100 0011

S_5: 0011 0110

S_6: 1100 1001

(if 1001) → 1st and 4th cols.

Q_S: 1011 0000

S_1: 0001 1110

$\wedge$ Q_S: 1011 0000

0001 0000 → not a match

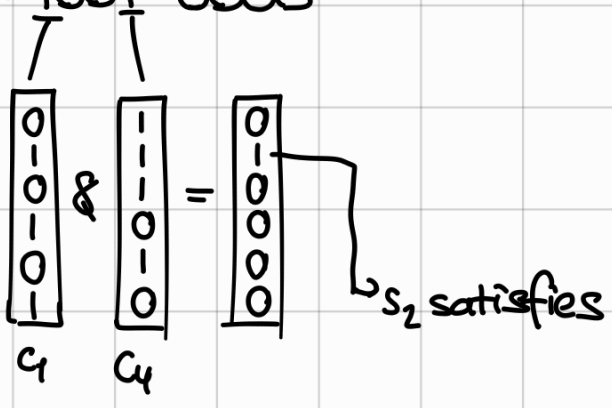
$\neq Q_S$

C1: 010 101

C2: 010 100 → Q_S: 1001 0000

C3:

C4: 111 010



Query weight:

1's in a query's Qs

↳ w(Q)

Ex. Q_S: 1001 0000

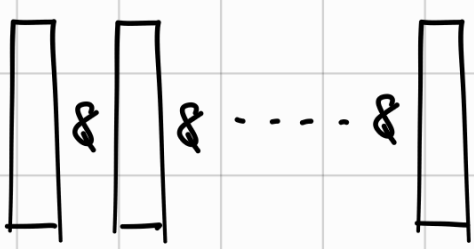
$w(Q_S) = 2$.

Paper:

Partial Evaluation of Queries for Bit Sliced Signature Files (Information Processing Letters, Kocaberber)

$$op = \frac{\#\ 1\text{'s in a signature}}{\text{signature length}} = \text{on bit density}$$

$$\text{false drop probability} = \frac{\#\text{matches}}{\underbrace{\#\text{signatures}}_{N}}$$



After processing i many bits of the query:

$$f_{d_i} = op^{(i)}$$

(this is I think for bits not the same i in d_i.)

ASK

so maybe revise the formula as $f_{d_j} = op_j^i$? after i^{th} bit

freq. of $d_i = s_i$ — on bit density of s_i

- We assume all matches are false matches.

$RT(i)$ = response time after processing i many bits

$$= i \cdot T_{slice} + N \cdot op^i \cdot T_{resolve}$$

↳ assume false match

$N \cdot op^i$ = # matching signatures

T_{slice} = time needed to process a signature

N = #sigs

To find the i value that minimizes the $RT(i)$:
(don't need to memorize the derivation)

↳ We take the derivative of $RT(i)$ wrt i:

$$\frac{d\,RT(i)}{di} = T_{slice} + N \cdot T_{resolve} \cdot op^i \cdot \ln op \quad (1)$$

To find the optimum i:

↳ Let the eq. (1) be 0 and solve for i:

$$op^i = \frac{T_{slice}}{N \cdot T_{resolve} \cdot (-\ln op)}$$

$$\ln op^i = \ln\left(\frac{T_{slice}}{N \cdot T_{resolve} \cdot (-\ln op)}\right)$$

$$\Rightarrow i = \ln\left(\frac{T_{slice}}{N \cdot T_{resolve} \cdot (-\ln p)}\right) / \ln(p)$$

$w(Q) = \#$ 1's in a query

if $i > w(Q)$ then $i = w(Q)$

[i.e. if $i = \arg\min RT(i) > w(Q)$, then stop preprocessing at $w(Q)$

else, preprocess until i (don't go through all $w(Q)$]

Horizontal Partitioning

(Lee & Leng, ACM TOIS, 1993)

Signature density = 0.5 (half $\underset{50\%}{}$ of the bits are 1)

S_1 : 0111 1000

S_2 : 1000 1011

S_3 : 0011 1100

S_4 : 1100 0011

S_5 : 0110 1100

S_6 : 1001 0011

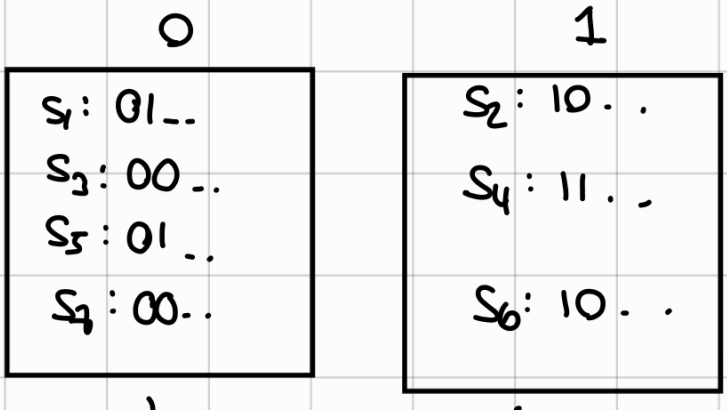
S_7 : 0000 1111

Q = 1001 0001

=

Fixed prefix partitioning

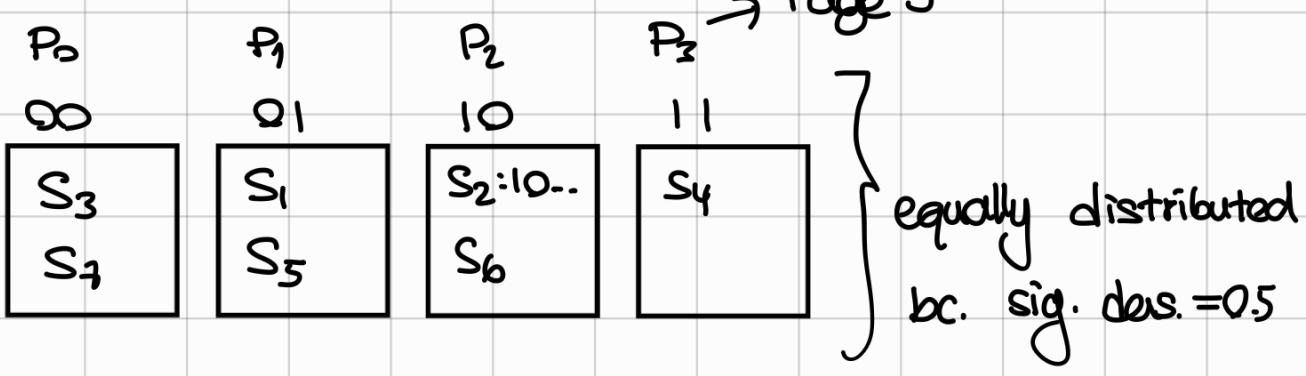
$k = 1$ (use 1-prefix):



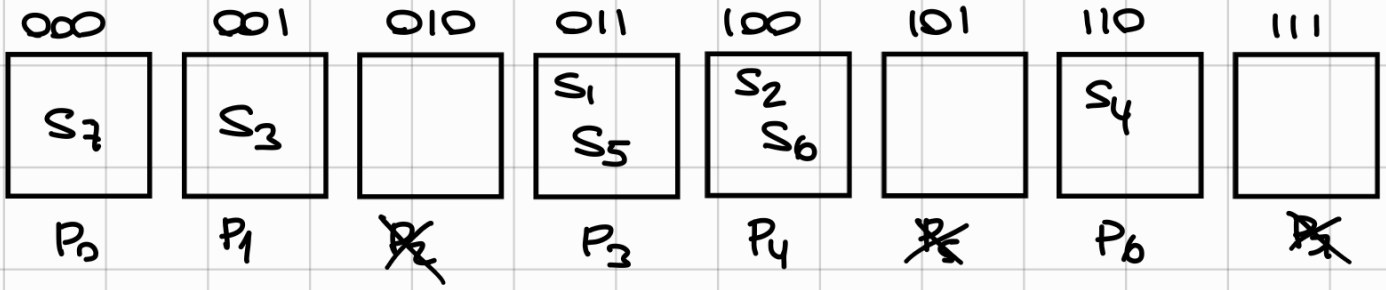
starts w/ 0 ↙ ↘ starts w/ 1

Query → Q = 1001 0001 ⇒ start's w/ 1

$k: 2$



$k = 3$ (Why not equally dist. now?)



Q (1001 0001) matching partitions: 100 → 1XX → P_{100}, P_{101}, P_{110}, P_{111}.

(X: don't care)

Access a page if Q_{sk} & $P_s = Q_s$, then access that page for false drop elimination.

If Q = 0000... we would have to access all the pages.

" " = 1110.. " " " " " 1 page.

⇒ # partitions to access: $2^{(\#\text{ 0's in the first } k \text{ bits of } Q_{sk})}$

Query	k=1	k=2	k=3
Q_1 = 1000...	1(1)	2(10,11)	4(100,101,110,111)
Q_2 = 111...	1(1)	1(11)	1(111)
Q_3 = 110...	1(1)	1(11)	2(110,111)

↑

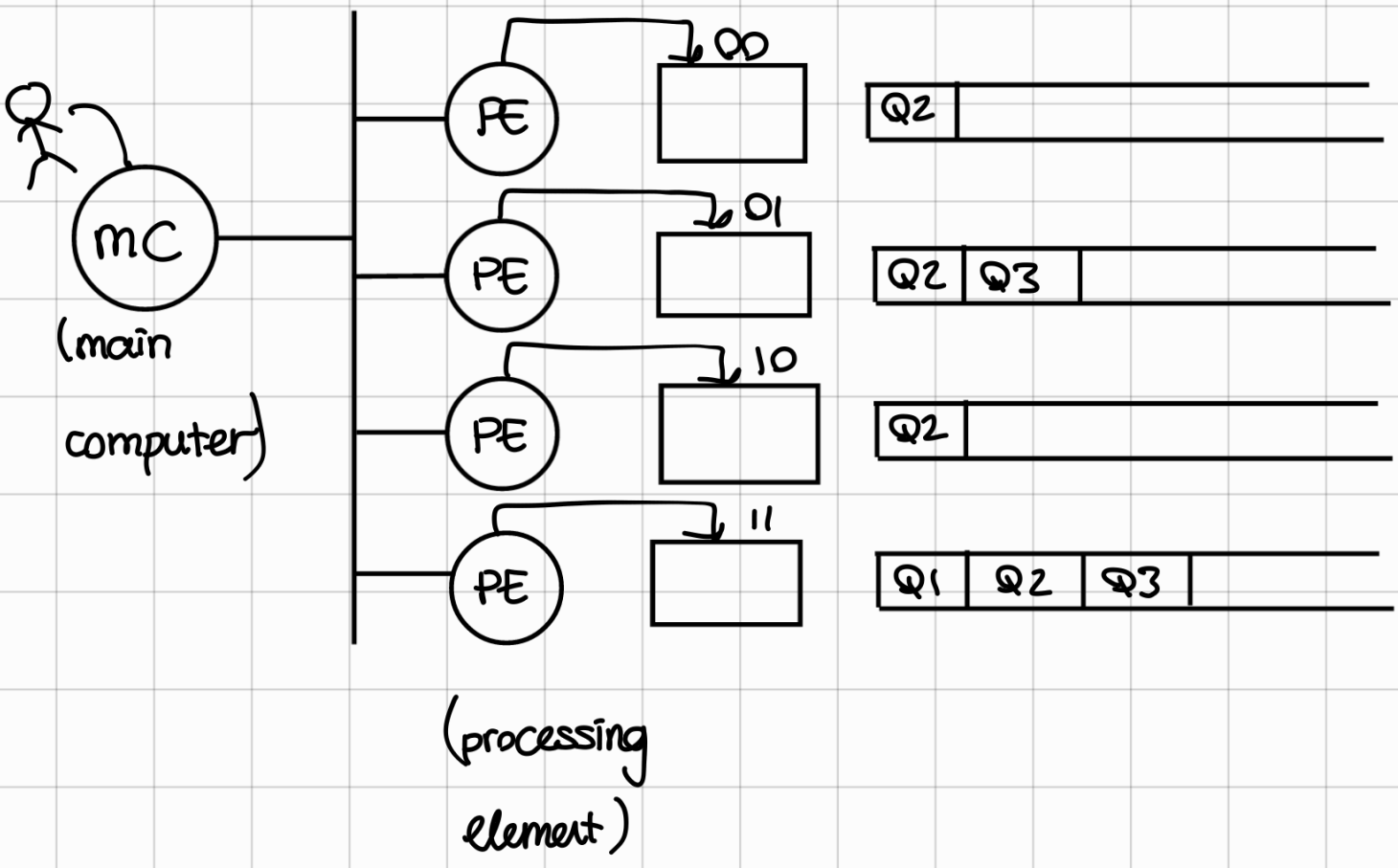
pages to be accessed (page no)

Connection Machine, paper, 1987

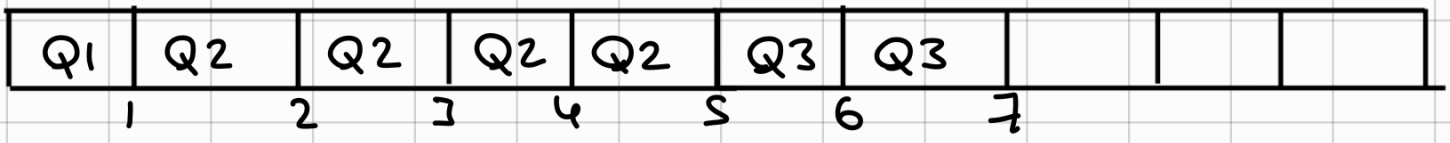
Parallel Environment

for $k=2$:

	$k(2)$
Q_1 : 1110 ..	1(11)
Q_2 : 0000 ..	4(00,01,10,11)
Q_3 : 0110 ..	2(01,11)

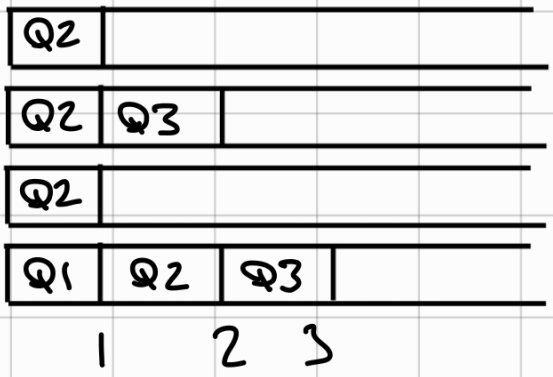


Sequential processing:



Q_1: 1 tu Q_2: 5 tu Q_3: 7 tu

Parallel processing:



Q1: 1tu tu: time unit

Q2: 2tu

Q3: 3tu

Avg turnaround time = (completion time - arrival time)

Seq. $\frac{(1-0)+(5-0)+(7-0)}{3} = 13/3 = 4.33$ tu

Par. $\frac{(1-0)+(2-0)+(3-0)}{3} = 6/3 = 2$ tu

Throughput: # jobs completed with time

Seq: $\frac{3 \text{ jobs}}{7 \text{ tu}}$ Parallel: $\frac{3 \text{ jobs}}{3 \text{ tu}}$

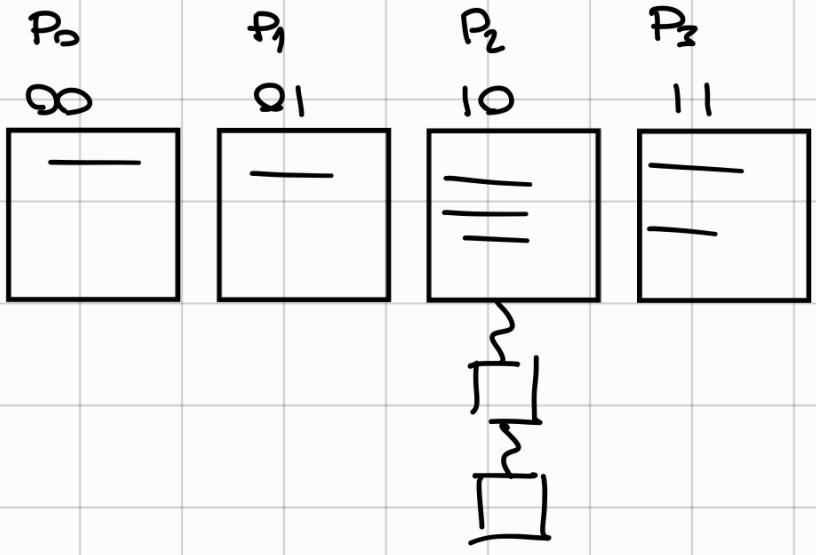
Speed up ratio: total seq time / total par time

7/3.

08.12.2023

Fixed Prefix

$k:2$



How many "# Qs vs sig. comparisons"?

$Q = 1001 \quad 0011 \quad k=2$ 1X: P_{10} and P_{11} will be accessed.

- There is an overflow problem:
  if ∄ place in the page then we have overflow pages. It slows the response time down.
- If we have too many overflow pages, we have to maintain the file.
  we want data structures that maintain themselves.

Extendable Hashing

- Extendable Hashing paper — Fagin et al, ACM TODS 19 } uses prefix
- Linear Hashing paper — Litwin et al, 1980 VLDB } uses suffix

} both horizontal partitioning

Example. S1 : 0011 1000

S2 : 0100 0011

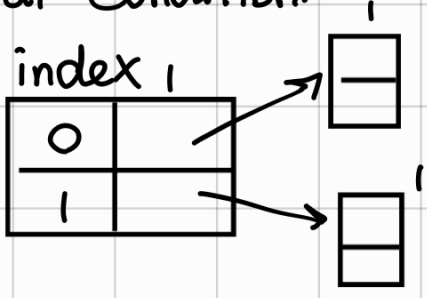
S3 : 0100 1011

S4 : 0100 1111

S5 : 0000 1111

Blocking factor = BF = 2.
||
2 signatures per block.

Initial Condition.

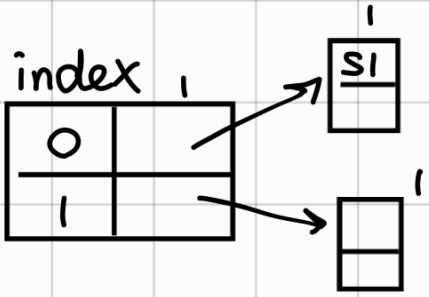


index = directory

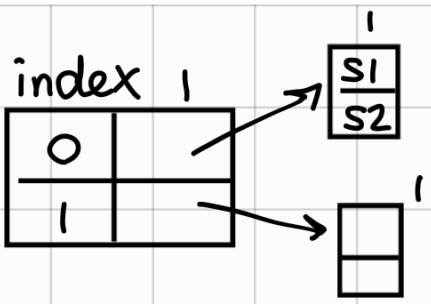
directory level = 1

page level = 1

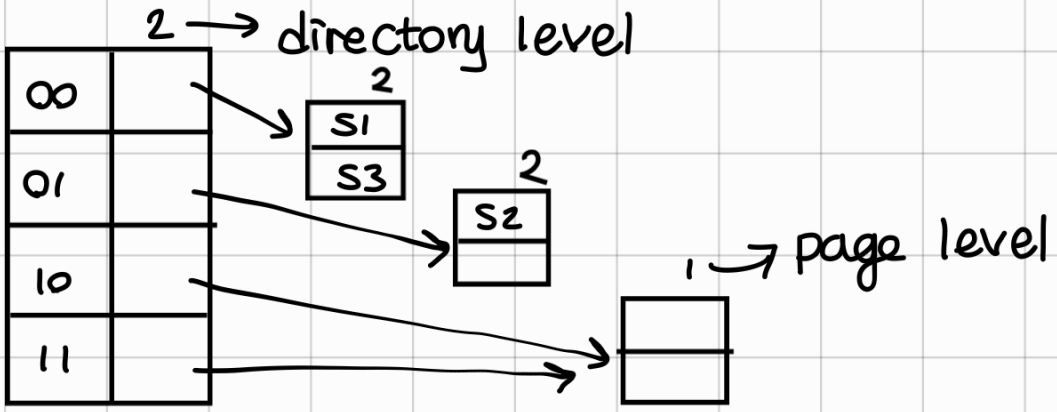
Insert S1 :



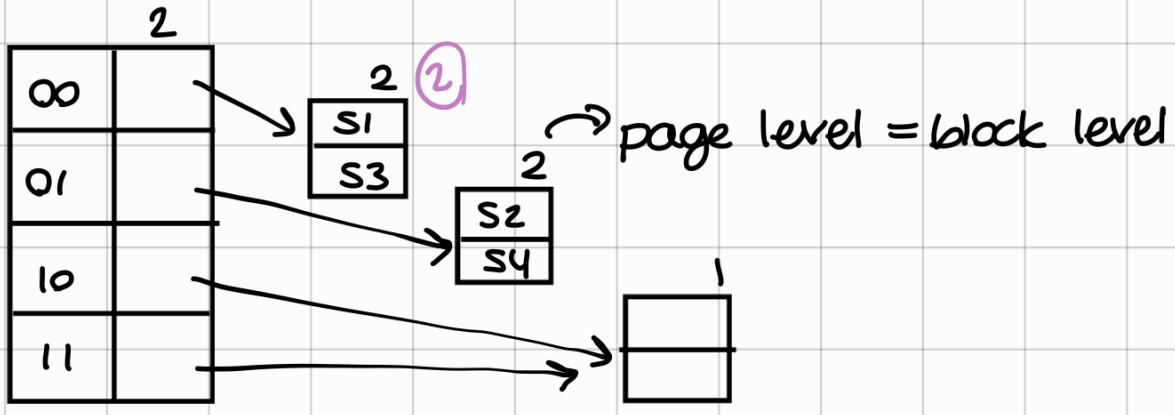
Insert S2:



Insert S3:



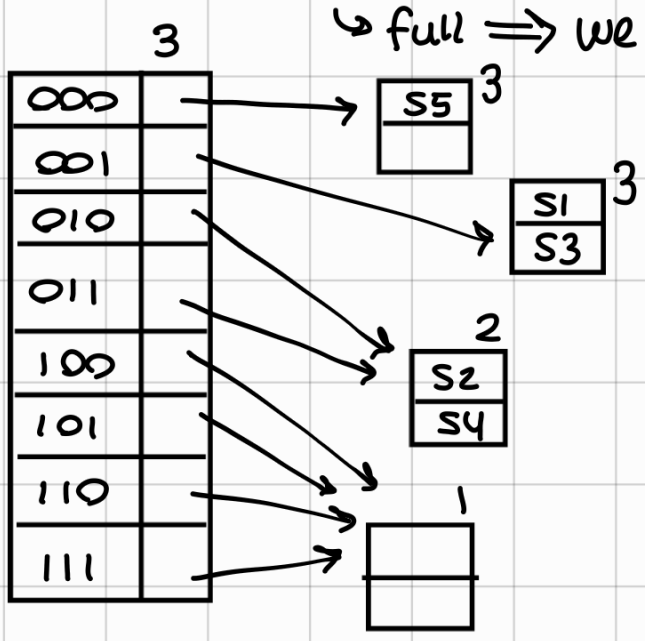
Insert S4:



- If the storage is critically low, you don't need to store the first 2 bits for the above structure. e.g. we know that the first 2 bits of S1 and S3 are "00".

Insert S5 : 0000 1111

↳ full ⇒ we need to increase the directory level.



$1 \leq$ page (block) level $\leq$ directory (index) level

Let Q = 0110 0000.

Directory level = 3 ⇒ k = 3. ⇒ Q = 0110 0000.

⇒ Access $Page_{011}$ or $Page_{111}$.

+ Extendible hashing prevents overflow pages!

vs

- There can be overflow in Linear Hashing 😢

+ We don't need to keep indices like directory level in Linear Hashing. 😊

Linear Hashing vs Extendible Hashing

both horizontal partitioning

Linear Hashing	Extendible Hashing
x	no overflow
no index keeping	x
suffix based	prefix based

12.12.2023

- Fixed prefix : dog, needs cleaning
- Linear Hashing, Extendable Hashing : cat, cleans itself

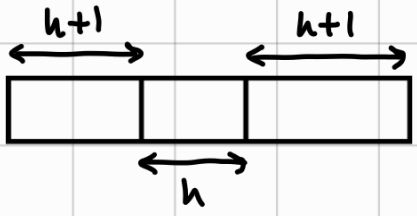
Ex. 0101 0011 2-suffix 11
3-suffix 011

1993 QuickFilter

Linear Hashing

Insertion Algoritlum

1. Use the h-bit suffix or h+1 suffix to find the page to insert.



LF = Load Factor = $\frac{\#\text{recs in file}}{\#\text{recs in prime area}}$

Bkfr = blocking factor = # sig.s that can be stored in a page

2. When LF is at the desired level after adding LF x Bkfr sig.s, add 1 more block to the file; and distribute the records of the block (pointed by bv) between this new block and the bue block.

3. Update bue (boundary value):

bv = bv + 1

if $bv = 2^h$ then

h = h + 1

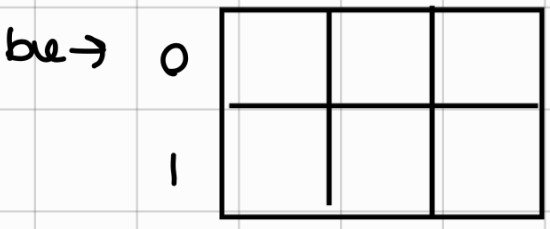
bv = 0.

Example.

S1: 0011 1000	S5: 0000 1111	S9: 0001 1100
S2: 0100 0011	S6: 1011 0001	S10: 1100 0100
S3: 0010 1011	S7: 0001 0101	S11: 1101 0100
S4: 0100 1111	S8: 0100 0000	S12: 1111 0010

Initially: $br = 0$, $Bkfr = 3$, $h = 1$, desired $LF = \frac{2}{3}$.

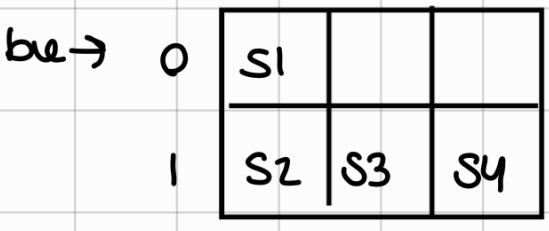
desired hashing level



Add $LF \times Bkfr = \frac{2}{3} \cdot 3 = 2$ blocks at a time.

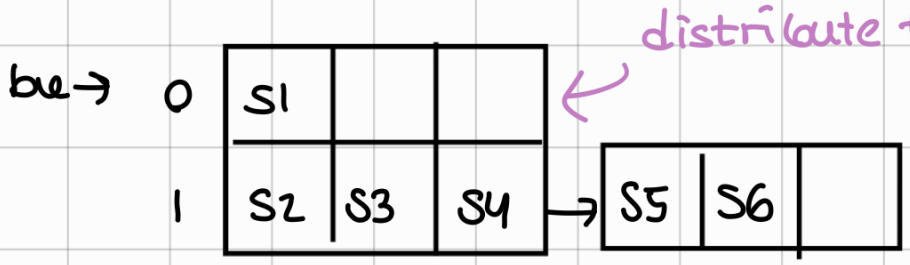
Insert S1, S2:

Insert S3, S4:



$LF = \frac{4}{6} = \frac{2}{3} \Rightarrow$ add 1 block & update by distributing

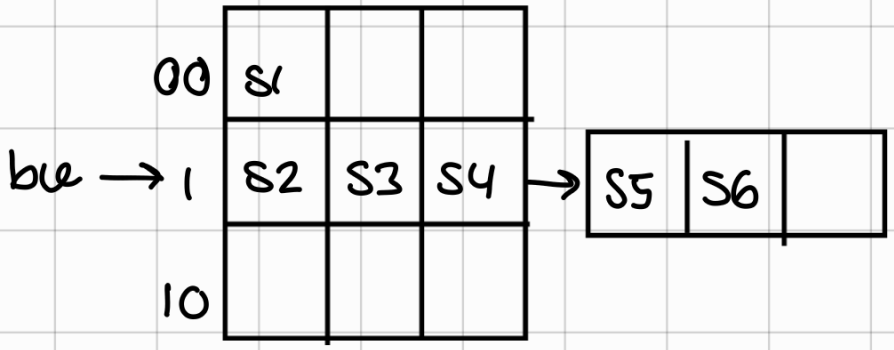
Insert S5, S6: → added 2 sig.s,



now update (the file & be)

be = be + 1 = 1.

$LF = \frac{6}{6}$ ✗

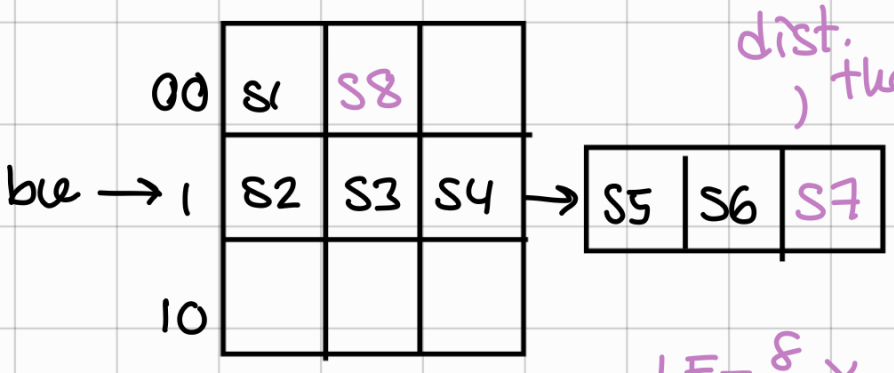


$LF = \frac{6}{9} = \frac{2}{3}$.

⇓

add 2 more

Insert S7, S8:



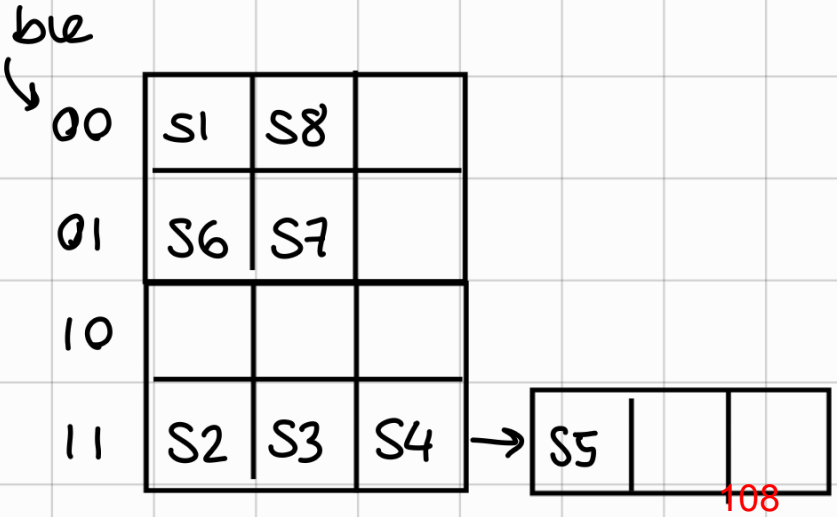
be = be + 1 = 2

$= 2^h = 2^1$

↳ be = 0

h = h + 1 = 2.

$LF = \frac{8}{9}$ ✗



$LF = \frac{8}{12} = \frac{2}{3}$ ✓

Insert S9, S10:

$LF = \frac{10}{12}$ ✗

$be = be + 1 = 01$

be →

00	S1	S8	S9	→ S10
01	S6	S7		
10				
11	S2	S3	S4	→ S5

– distribute these into 000, 100

Update file:

000	S1	S8		
be → 01	S6	S7		
10				
11	S2	S3	S4	→ S5
100	S9	S10		

$LF = \frac{10}{15}$ ✓

Insert S11, S12:

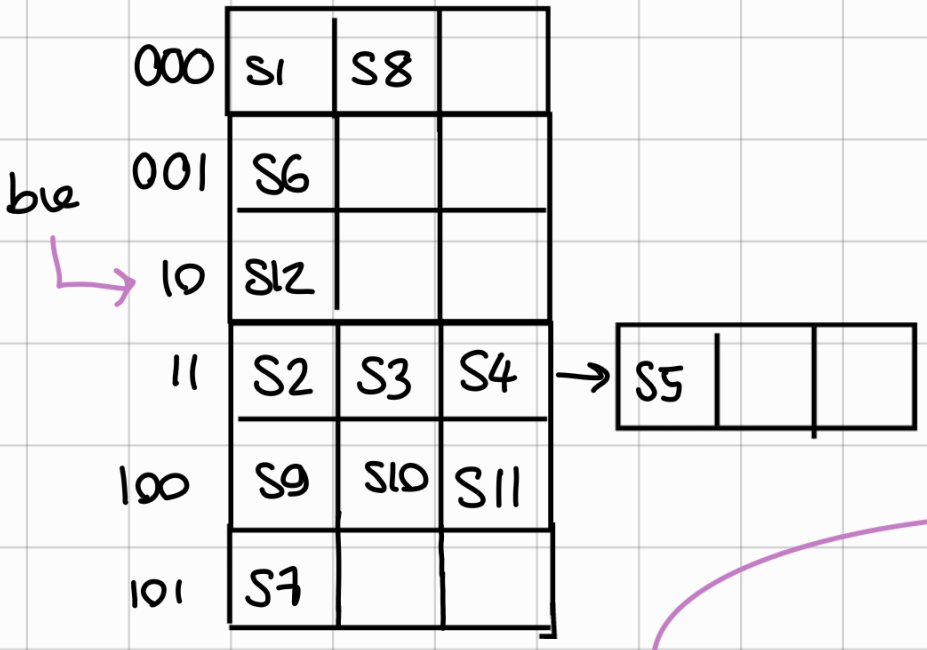
$be = be + 1 = 10.$

$h = 2.$

000	S1	S8		
be → 01	S6	S7		
10	S12			
11	S2	S3	S4	→ S5
100	S9	S10	S11	

$LF = \frac{12}{15}$ ✗

Distribut 01 into 001 & 101 to maintain LF.



ble = ble+1.

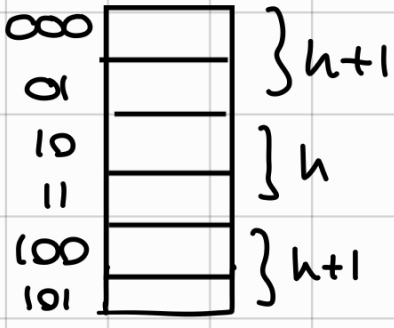
= 01+1 = 10.

LF = 12/18 ✓

Insert 2 sigs, if LF ✓

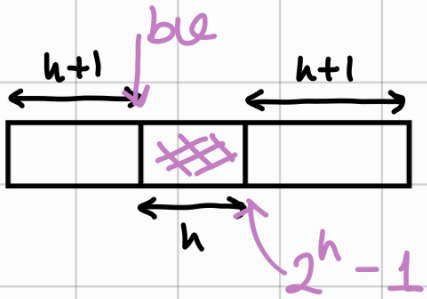
if LF > desired

dist. & update ble.



if we add a new sig, the structure would change

- n = # pages → n = 6
- h = $\lfloor \log_2 n \rfloor$ → h = $\lfloor \log_2 n \rfloor$ = 2 ✓

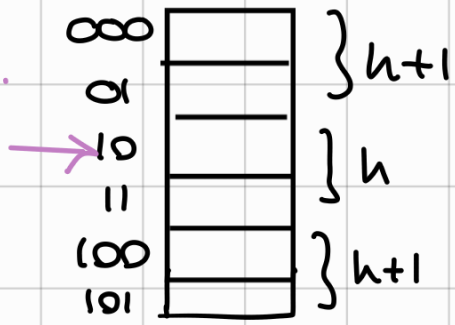


Pages $[ble, 2^h - 1]$ are at level h.

- # pages at level $h+1 = 2 \cdot ble$
- # " " " $h = n - 2ble$

- $ble = f(n,h) = n - 2^h$

ex.



$n = 6$

$h = \lfloor \log_2 n \rfloor = 2.$

$ble = n - 2^h = 6 - 4 = 2 = 10$

pages at level $h+1 = 2 ble$
$= 4.$

Query Processing

Q: 0010 0110

if $Q_k \wedge P_k = Q_k$

access page

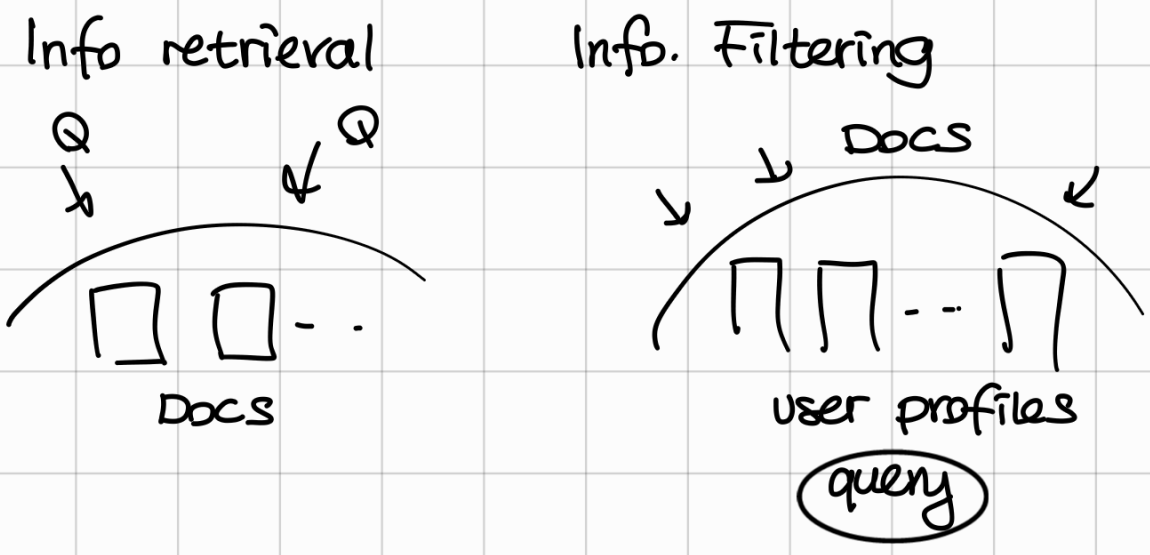
P_k	Q_k	$P_k \wedge Q_k$
000	110	000 ✗
01	10	00 ✗
10	10	10 ✓
11	10	10 ✓
100	110	100 ✗
101	110	100 ✗

19.12.2023

Info. ret. & info. filtering : Two Sides of the same Coin

Bruce Croft UMass 1993

Belkin Rutgers CACM



One last approach in information filtering systems smth like; new event detection tracking

The filter bubble : how you are trapped in an echo chamber of your own interests, and not challenged

ACM TODS, TW Yang, Hector Garcia Molina:

- conjunctive user profiles: a doc satisfies the user profile if it has all of the terms in the profile.

1. Brute Force Approach

a) Compare the incoming doc. with each profile one by one. (Random Brute Force)

b) Use an occurrence table (Ranked Brute Force)

Random brute force	terms that appear in very small # of docs.
Ranked brute force	terms appear in many docs.

2. The Count Method

Sample profiles

P1 = (a, b) → user P1 is interested in terms a and b.

P2 = (a, d)

P3 = (a, d, e)

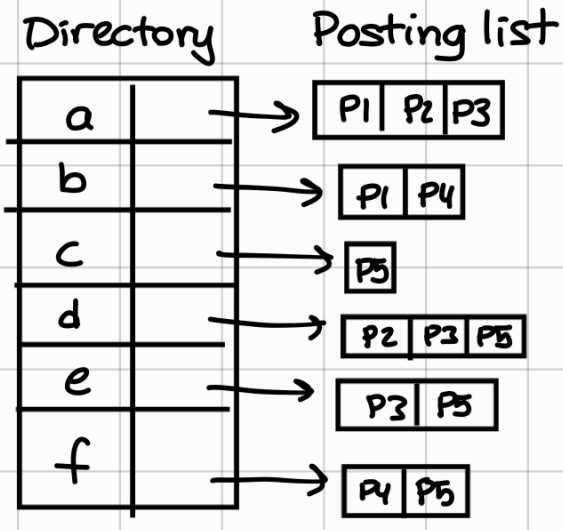
P4 = (b, f)

P5 = (c, d, e, f)

Unique doc terms = {a, b, c, d, e, f}

sample doc = a, c, a, f, b, c

inverted index → structure for profiles



size of each profile in terms of the # terms it has

↳ TOTAL

	TOTAL	Count Array a	b	c	f
P1	2	0	1	2	2
P2	2	0	1	1	1
P3	3	0	1	1	1
P4	2	0	0	1	1	2
P5	4	0	0	0	1	2

sample doc's terms = {a, b, c, f}

Total (P1) = Count Array (P1) } This doc satisfies P1
Total (P4) = Count Array (P4) } and P4.

Total (P5) > Count Array (P5) → P5 is not satisfied w/ doc: a, c, a, f, b, c

3. The key method

In the key method a profile only appears in one of the posting lists. That term is called the key.

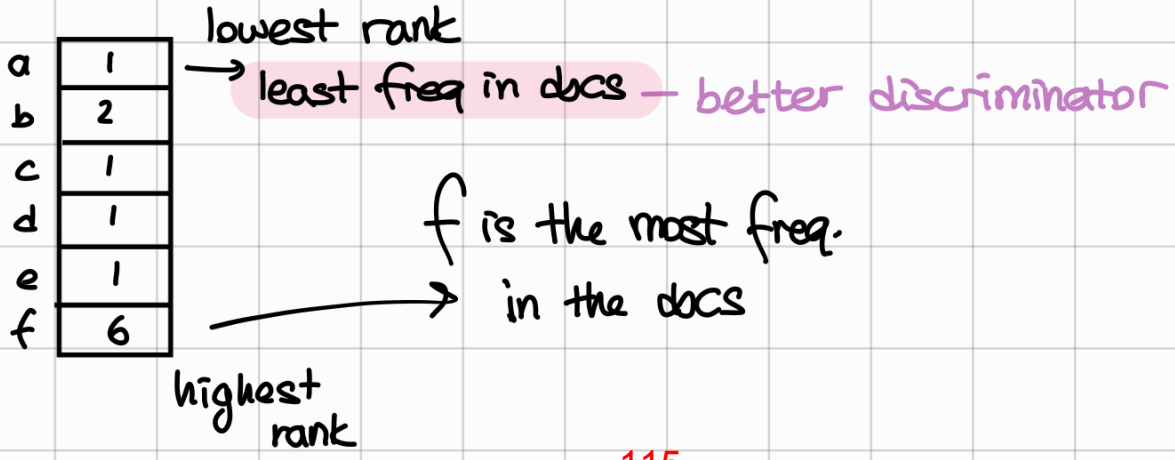
(∃! term, i.e. key, whose posting list has the profile.)

a. Random key

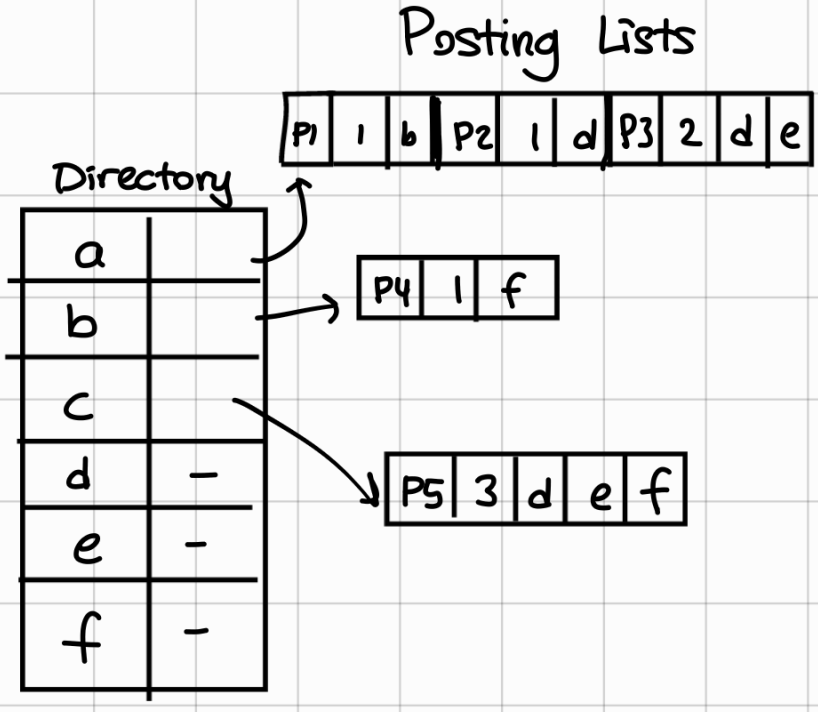
Any word is the key

b. Ranked key

Store the profile in the list of the word w/ lowest rank



- A user profile can appear in only one posting list.



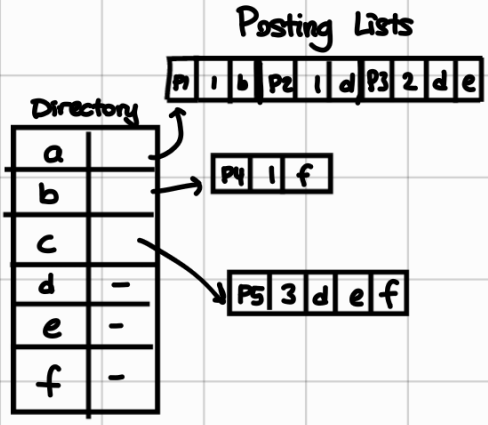
Doc : a, b, c, f

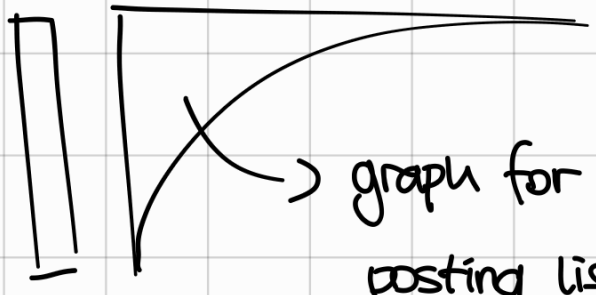
Consider posting list of a (is in doc): P1(b) P2(d) P3(d)

b: P4(f)

c: P5(d,e,f)

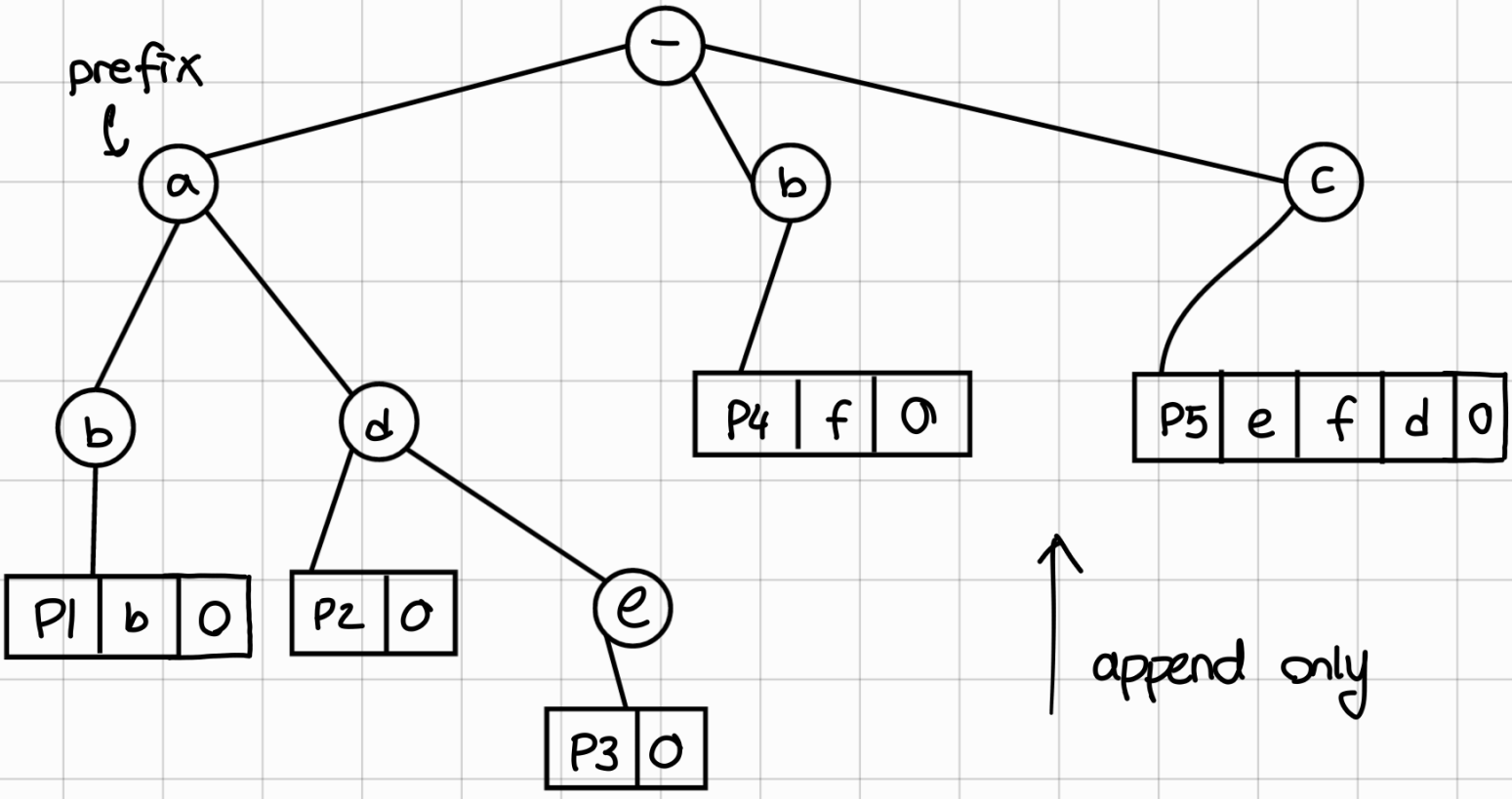
} doc satisfies P1 & P4





→ graph for # elems in posting list. It polynomially decreases as the freq of the term increase.

4. The Tree Method



- Tree structure saves space when ∃ many common prefixes among the profiles.

⚠ These methods are only for conjunctive queries

Notes are by Ecem İlgün
From Fall 2023