12/02/2008

Information Retrieval

Books (Available on the Web)

Information Retrieval , Van Rijsbergen Univ of Glasgow

× Information Retrieval : Algorithms & Data Structure
Chp8 ?. W. Frakes & R. Baeza-Yates.

Modern Information Retrieval : R. Baeza-Yates
N. ... ?

⇒ acm.org/dl

Conf. ACM SIGIR Acm SIKM
Into. Ret. Information & Knowledge management

ACM Trans on Information Systems
" " " Database Systems
Journal of the American Society for Information Science & Technology (JASIST)
Information Processing & Management (IPM)

New Event Detection & Tracking

⇒ Efficient & Effective
↓
in terms of time & space
collect
↳ 400 000 dif. news → determine new events

2007 → 9 new resources ⇒ Hürriyet, Milliyet ... (Collection)

Midterm ⇒ 25 March 2008 ⇒ Tuesday

Final ⇒ 21st may - 31st may

ad hoc queries ⇒ word .. word ...
queries ↗
they do similarity comparison
rank from most similar to the less similar

abundance problem

calculate similarity between queries.

Efficient
effective system → relevant documents at the top

System Evaluation:

Recall & Precision

TREC : Text Retrieval Conference.

trec eval package provides performance

bpref : binary preference

there is a part of data which has not been evaluated by human being & assumed as irrelevant.

Clustering & Cluster Validation:

grouping.

tf idf → inverse document frequency
term frequency

a word appears in all of the documents ⇒ bir,ve -
if a term appears smaller number of times ⇒ can be used for differentiation

(fundamental file structures)

information → (d10, d11, d30, d5) ← posting list

retrieval → d1, d10, d51

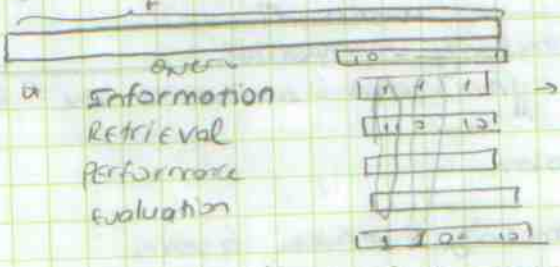
Query Processing:

Acm Computing, surveys, Alistair moffat, Justin Zobel,

Inverted Files for text search engines

Signature Files:

For each document we have a bit array, refered to as signature:



F : size of the array
→ set random location 1
y not value random
Random Number Generator
with a seed

then we superimpose this ⇒ order this columnwise ⇒ OR ing

Query Signature. if $(Q_S \& D_S) = Q_S$
Document " then document is retrieved.

{ if (Qs & Ds) = Qs
then document is retrieved }

↳ false drop resolution

assume you're dealing with huge data

↳ So have to make this more efficient

N-GRAM

2gram 3gram Information : in, nf, fo, or, rm, ma, at, ti, io, on
bigram trigram

in * on ⇒ word begins with in & ends with on.

so * m * t

PAT TREES: (Patricia Trees)

(Suffix Trees) trei

from sniffer

good for intelligent purpose

similar to priority queue.
record everything in telephone conv
If you construct pat tree you can get anything.
any string

for movies ⇒ pat tree structure you can easily find it (Not so easy to construct)

Compression:

Assume we have posting list

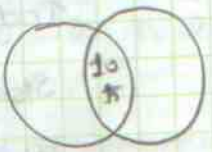
posting list

10	5	2	5	3	3	2
information → 10, 15, 17, 22, 25, 32, 34, 42, 47						

⇒ the #s of the doc that the word appears

25 42

retrieval → 10, 15, 20, 30, 47 } this form of the
→ 27, 32, 42 ... then



But the numbers will increase → the digits will ↑

d-gap concept ⇒ by doing so digits ↓
↳ document we make the #s
↳ efficient

10 + gap1 = 15
15 + gap2 = 17

say for some reason we begin with retrieval cause it's smaller

with skip first begins with 27
with
& the skipping (information) ⇒ we know we can skip first 5 of them since 25 < 27

Self-Indexing
Moffat & Zobel

ACM, TOIS, 1996
93.

INFORMATION RETRIEVAL SYSTEMS OVERVIEW:

Information Systems:

DBMS — structured data — Precise Answers

IR — unstructured data — ≈ answers

Traditional IR Systems:

MEDLARS provided by national Institutes of Health

MEDLINE online version of MEDLINE

LEXIS/NEXIS law

STAIRS provided by IBM

1993-1994

AltaVista

⋮

Google

SMART ← research IR system

Cornell Univ.
Gerard Salton

Components of An IR System

User Interface

Indexing System

Storage System (the file system that we want to use)

Query Processing System

Zobel & Moffat

↓

Inverted index for text search engines

An Indexing Example:

Doc 1: Information Retrieval by Parallel Document Ranking
Doc 2: An analysis of parallel text retrieval systems
Doc 3: Information retrieval in the law office: an overview

Stopwords: Frequent words (not good for distinguishing documents from each other)

stopword list ≈ 300 { the, and, a, an, this, ⋮

Cornell SMART forum (Christ for)

say we use Dx 3

Stopword list:
an
by
in
at
overview
systems
the

Indexing Text
Information
Retrieval
:
law
office

* uncontrolled indexing environment:
: include any word that is in the document or long as it doesn't appear in the stopword list

* Controlled Indexing
the indexing terms will be provided before hand

MeSH : Medical search Hierarchical ?

after a while any word encountered may have been used before

(full of 0's * of 1's are low)

Indexing Terms in Alphabetical Order:
1) analysis
2) document
3) information
4) law
5) office
6) parallel
7) ranking
8) retrieval
9) text

(sparse matrix)
binary matrix

$n = 9$ (no of words)
$m = 3$ (no of Documents)

$n \times m$
9×3

$$D = \begin{matrix} \\ d1 \\ d2 \\ d3 \end{matrix} \begin{bmatrix} t1 & t2 & t3 & t4 & t5 & t6 & t7 & t8 & t9 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$d_{ij} = (1 \le i \le m,\ 1 \le j \le n) = \begin{cases} 1, & \text{if } t_j \text{ appears in } d_i \\ 0, & \text{otherwise} \end{cases}$$

INSPEC database (on line)

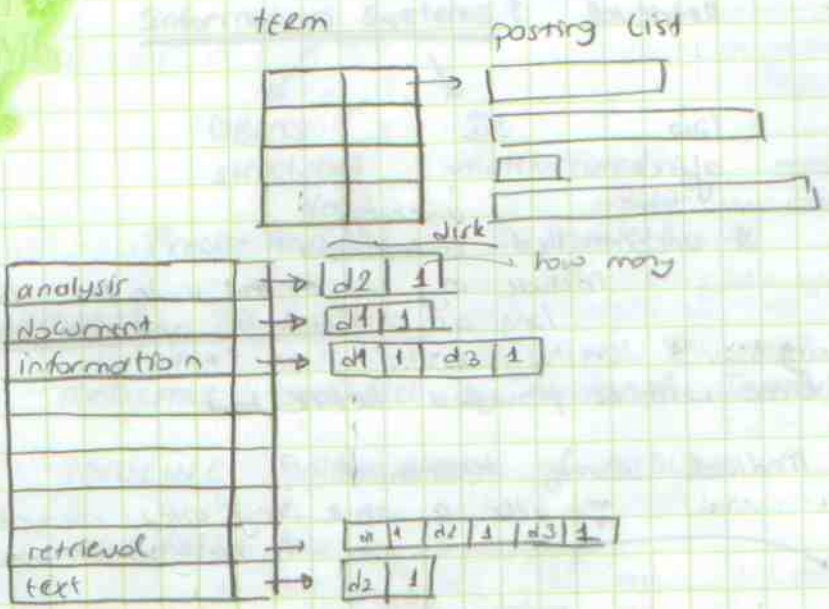
David Lewis

ACM,
Time
MPL

m : 12684
n : 14573

D matrix of inspec : density of 1's $\approx 2\%$

Inverted Index for the Example Collection:



Assume a Boolean Query Environment

Q: information & Retrieval

19/02/2008

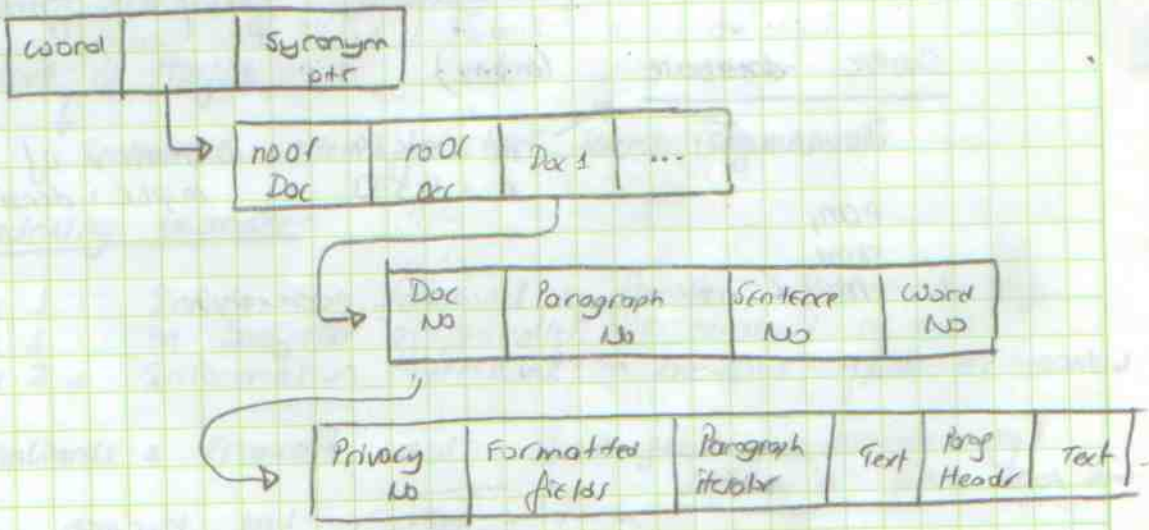
Example IR Systems:

STAIRS: Storage Information Retrieval Systems

Designed for mainframes, IBM

File System / Structure

Dictionary



Formatted Field:

Author Name
Journal Name
Page No
⋮

STAIRS has two programs:

1. utility programs : for database initialization & maintanance
2. Query & Retrieval Utility Systems (ASQUARIO)

Modes Of Operations:

Search Mode : for textual IR

Select Mode : for structural IR
↳ using formatted fields

Queries:

HEART
HEART OR DISEASE
HEART & and DUEASE & 3
WITH : in the same sentence
SAME : in the same paragraph

- Find matching documents using a boolean query
- Rank matching documents according to their significance

Value of a query term : $f(a, b, c)$

a : the frequency of a word in the document

b : retrieved set

c : the no. of documents in the retrieved set in which the term occurs

Query Term Value : $\frac{a \times b}{c}$

Score of a document = $\sum$ value of all query terms which appear in this document

Evaluation Of IR Systems:

users point of view
implementors point of view) ← System Resources
↳ effectiveness

efficiency : response time
system resource requirement x

- coverage
- Links should be alive
- ease of use

How to measure effectiveness?

TREC : Text Retrieval Conference [1992, ...]

← TREC 3 Appendix A

www.nist.gov/trec ? (twik) → Wikipedia

↳ national institute standards of technology

trec_eval package

PRECISION (p) = # of retrieved & relevant documents / # of retrieved documents

✓
x
x
x
✓

$\frac{2}{5} = 0.40$ — p10 p20 precision @ 10 or 20

↳ just one // two pages

RECALL: # of retrieved & relevant documents / (total # of relevant docs in the collection)

$\frac{2}{20}$ → total # of rel. doc

TEST COLLECTION IN IR

A set of documents
A set of queries & their relevant documents) ≥ 50

≥ in statistical sense 50 ;)
(if you get 50 relevant enough)

Standard test collections facilitate repetition of tests.

Finding relevant documents for queries:

TREC uses the pooling concept

Retrieve top 100 documents & identify the relevant ones for query

← if all distinct = 1000 docs

not all distinct ☹ 700 docs

all other docs they will be assumed as irrelevant

How reliable?

ACM SIGIR 1998
Justin Zobel

2,2741

Documents which are not seen by the evaluators (annotators) are assumed as irrelevant
Actually some of them can be relevant

A new measure (ACM SIGIR Conf. 2004, by Buckley & Voorhees)

bpref : binary preference

MAP : bpref is as reliable as MAP

↳ (mean Average Precision)

$$MAP = \frac{P_1 + P_2 + P_3 + 0}{4}$$

All together there are 4 relevant docs

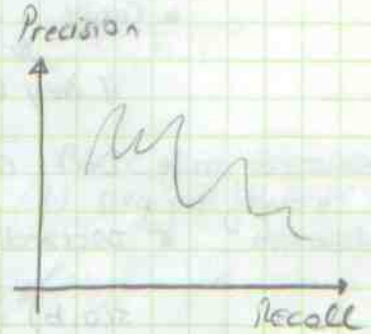
Example for Recall & Precision

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

Dayarthik

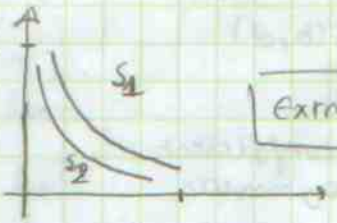
Ahmed

total no of relevant docs = 10



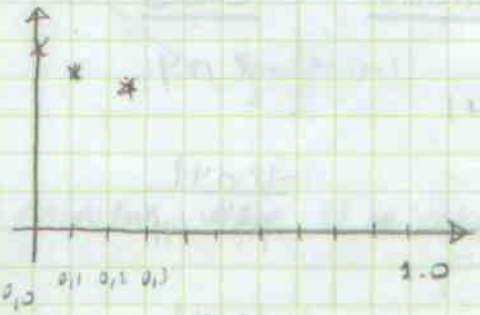
more relevant
precision ↓

S1 is better than S2



Extrapolation ↗ to prevent those big tags

TREC



makes the curve nicer...

11 point

t test → available

SNO	MAP1	MAP2
1		
2		
.		
i		
50		

Is the difference statistically significant

Student t-test

2-tail

1-tail

not only looks at avg compares the individual values gives a "p" value

→ p = 0,05

if p value is p < 0,05 significant

0.45 0.55

Similarity Calculation:

MOTIVATION: - For ranking documents according to their similarity to submitted query

browsing

- Cluster documents according to similarity to each other then use these clusters to find additional relevant documents like your favorite document

* There are several similarity coefficients

Most of them is symmetric $\Rightarrow s(a,b) = s(b,a)$

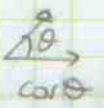
Van Rijsbergen, Information Retrieval, Univ. of Glasgow

Simmilarity Coefficient

↙ Binary

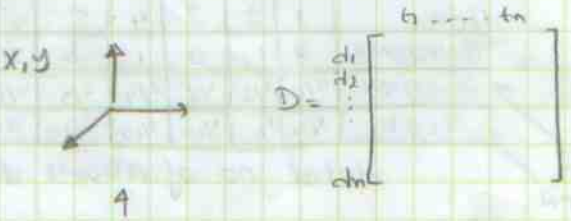
↘ Weighted

* Dot product
* Cosine



if they overlap $\Rightarrow \cos 0 = 1$

* Dice Coeff
* Jaccard



n- dimensions

$s(a,b) = s(b,a)$

cover coefficient
symmetric

Simmilarity Coefficient	Binary	Weighted
Inner Product (Dot Product)	$X \cap Y$	$\sum x_i y_i$
Cosine	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert^{1/2} \lvert Y \rvert^{1/2}}$	$\frac{\sum x_i y_i}{\sqrt{\sum x_i^2 \sum y_i^2}}$
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	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert - \lvert X \cap Y \rvert}$	$\frac{\sum x_i y_i}{\sum x_i^2 + \sum y_i^2 - \sum x_i y_i}$

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

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

Inner Product $= 2$

Dice $= \frac{2 \cdot 2}{4+3} = \frac{4}{7}$

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

Cossine $= \frac{2}{\sqrt{4}\sqrt{3}}$

⇒ tf idf (we assign higher values to words which appear more frequently in documents ⇒ like stopwords)

↓

inverse document frequency

Salton Buckley Term Weighting Approaches Information Processing and Management. 1988

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

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

Dice $= \frac{2(2+0+2+3+10)}{(4+0+1+9+4)+(1+5+4+1+25)} = 0{,}69$ $\quad$ Cosine $= \frac{17}{(18.31)^{1/2}} = \frac{17}{23.6} \simeq 0{,}72$

Clusty

How To Calculate Simmilarity Among Documents

$$D = \begin{matrix} & t_1 & t_2 & \dots & t_n \\ d_1 \\ d_2 \\ \\ d_m \end{matrix} \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

$S_{ij} = S_{ji}$

Brute fra approach

1.) straight forward approach

$$S = \begin{matrix} & 1 & 2 & 3 & & m \\ 1 \\ 2 \\ \\ \\ m-1 \\ m \end{matrix} \begin{bmatrix} 1 & S_{12} & S_{13} & \dots & S_{1m} \\ - & 1 & S_{23} & \dots & S_{2m} \\ & & 1 & S & \\ & & & & S_{m-1} \\ & & & & 1 \end{bmatrix} \leftarrow m-1$$

Total no of simmilar value to be calculated $= 1+2+\dots+(m-1) = \frac{m(m-1)}{2}$

2) Using the knowledge of term distributions in documents

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

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

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

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

Consider d_1 :

t_1 t_2 t_5

$\{d_1, d_2\} \cup \{d_1, d_2\} \cup \{d_1, d_2\} = \{d_1, d_2\}$ $\quad s_{12}$

$$\begin{bmatrix} 1 & s_{12} & s_{13} & s_{14} & s_{15} \\ & 1 & s_{23} & s_{24} & s_{25} \\ & & 1 & s_{34} & s_{35} \\ & & & 1 & s_{45} \\ & & & & 1 \end{bmatrix}$$

d_2 :

t_1 t_2 t_4 t_5

$\{d_1, d_2\} \cup \{d_1, d_2\} \cup \{d_2, d_5\} \cup \{d_1, d_2\} = \{d_1, d_2, d_5\}$ $\quad s_{25}$

d_3 : t_6 $\{d_3, d_4, d_5\}$ $\Rightarrow s_{34}, s_{35}$

d_4 : t_3, t_6

$\{d_4, d_5\} \cup \{d_3, d_4, d_5\} = \{d_3, d_4, d_5\}$ $\quad s_{45}$

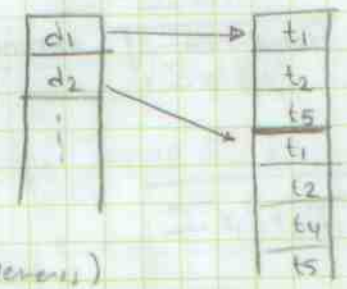
There are 10 similarity values to be calculated, but we need to calculate only 5 of them.

5/10 $\Rightarrow$ 50% saving

3) Using the Inverted File for Term

$d_1 \to$ | t_1 | t_2 | t_5 |

$d_2 \to$ | t_1 | t_2 | t_4 | t_5 |



use pointers to find the similarity → (see common elements)

in order to get rid of incorrect pairs

t_1 to t_1 of the same ✓ increment counter

t_2 t_2 " " " ✓ " "

t_5 t_4 " " " — increment pointer

t_5 t_5 " " " ✓ " counter

	t_1	t_2	t_3	t_4	t_5	t_6
d_1	1	1	0	0	1	0
d_2	1	1	0	1	1	0
d_3	0	0	0	0	0	1
d_4	0	0	1	0	0	1
d_5	0	0	1	1	0	1

docum, freq

$t_1 \rightarrow \langle 1,1 \rangle \ \langle 2,1 \rangle$ $t_4 \rightarrow \langle 2,1 \rangle \ \langle 5,1 \rangle$

$t_2 \rightarrow \langle 1,1 \rangle \ \langle 2,1 \rangle$ $t_5 \rightarrow \langle 1,1 \rangle \ \langle 2,1 \rangle$

$t_3 \rightarrow \langle 4,1 \rangle \ \langle 5,1 \rangle$ $t_6 \rightarrow \langle 3,1 \rangle \ \langle 4,1 \rangle \langle 5,1 \rangle$

Dice Coef : $\dfrac{2|X \cap Y|}{|X| + |Y|}$

document length array →

d_1	d_2	d_3	d_4	d_5
3	4	1	2	3

$$S = \begin{bmatrix} 1 & S_{12} & S_{13} & S_{14} & S_{15} \\ - & 1 & S_{23} & S_{24} & S_{25} \\ & & & & \\ & & & & \end{bmatrix}$$

Consider d_1:

Simmilarity Array:

S_{11}	S_{12}	S_{13}	S_{14}	S_{15}
X	~~0~~	0	0	0

← mail boxes

t_1, t_2, t_5 ~~1~~ ~~2~~ 3

$\dfrac{2 \times 3}{3+4} = \dfrac{6}{7}$

Consider d_2:

S_{21}	S_{22}	S_{23}	S_{24}	S_{25}
X	X	0	0	~~0~~

1

$\dfrac{2 \times 1}{3+4} = \dfrac{2}{7}$

t_1, t_2, t_4, t_5

1,1 2,1   1,1 2,1   2,1 5,1   1,1 2,1   } ignore (1,1 2,1) } increment others

Consider the Computation Requirements:

m

X_d : depth of indexing (no of avg. terms / doc)

t_g : term generality (avg. posting list length, avg. no of docs/term)

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

$t_4 \rightarrow \langle 2,1 \rangle \langle 5,1 \rangle \longleftrightarrow \langle 5,1 \rangle \langle 2,1 \rangle$

try to reduce the comparison looking at this order

En baştakiler yap X

kendisine gelene kadar sondan bakıyoruz

* Doing the calculations faster: organize the posting lists in reverse order: first higher numbered documents

$$x_d\, tg\, \frac{n-1}{n} + x_d\, tg\, \frac{n-2}{n} + \dots$$

$$x_d\, tg\, \frac{1}{n}\left(\frac{n(n-1)}{2}\right) = x_d\, tg\, \frac{n}{2}$$

CLUSTERING: Jain, Murty, Flynn Data Clustering
ACM Computing Surveys, Sept 1995

Clusty
Sophia

Read (link on the course web page)

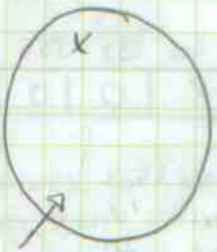
How to use clusters for IR:

1- For clustering the search results

cluster 1: information retrieval cluster 2: operating systems	

2- For browsing:

x → interesting document



look at the other members of this document's cluster.

3- Cluster-based retrieval

Salton & Students

C.T. Yu

- First choose the best matching clusters
- Then match your query with the documents of these selected clusters

Clusters → cluster centroids
Documents → document vector

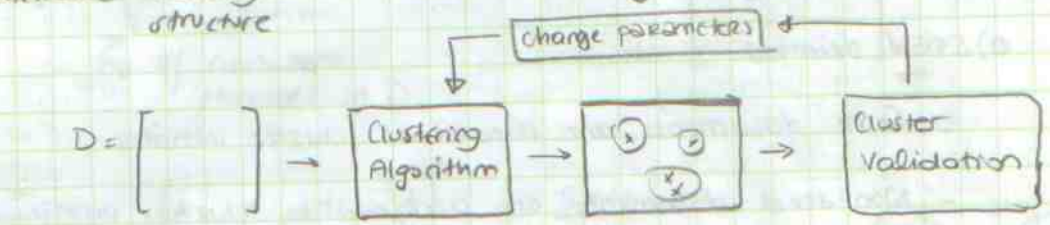
Classification of Clustering Algorithms

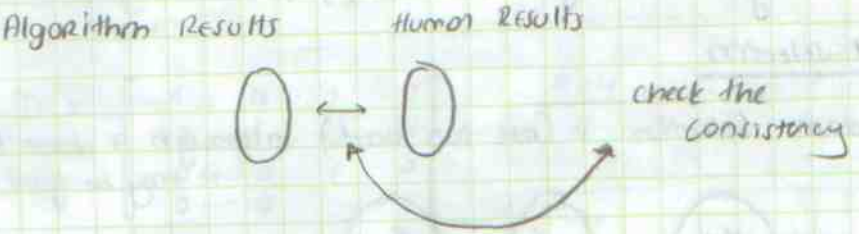
Categorization : groups are defined before categorization ← supervised classification

Clustering : groups are undefined when we begin ← unsupervised

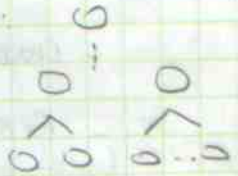
According to the generated clustering structure

According to its working principles





- partitioning : $C_i \cap C_j = \emptyset$, $i \neq j$
- overlapping : $C_i \cap C_j \neq \emptyset$, $i \neq j$ can be
- hierarchical :



- single_pass :
- multi-pass :
- graph-theoretical
- based on user queries

Omiciensti : ACM TODS, 1990

Peter J. Denning ⇒ (working set model'i çıkaran amca)

↳ IT Profession diye bir column'ı var.

Mastering the mass

Abstraction ← kasmadan ufak ufak (ingiliz ama yapmış)

Flow ⇒ sevdiğimiz bir işi yaparken nasıl da akıp geçer zaman

Mihaly Csikszentmihalyi

Communications of ACM April 2007

✱ Cluster hypothesis (Van Rijsbergen)

Functional Description: what ← easy

Operational Description: how ← difficult

Paul Erdös
Jain & Dubes
Clustering Algorithm
1988

1 ✱ Single-Pass Algorithm:

a) Seed oriented

- Some documents are selected as cluster initiates
- Non-seed documents are assigned to clusters initiated by seeds

? How many number of clusters (n_c) ?

✱ Seed-selection

- Random Selection (not too bad) although it doesn't seem so it may be good for some applications

- Choose first n_c
- Generate n_c synthetic seed documents
- Use posting list (Peter Willet)

take the first make a cluster
get the second if similar
put it in the first one's cluster
or it creates its own cluster
take third is it similar to any
of them ... and so on.

28/02/2008

Anderberg : 1973
Jain & Dubes : 1988
Jain, Flynn, Murthy: 1999

Single-Pass Algorithms : (continued)

a) Seed oriented

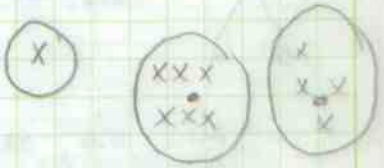
How to choose seed.

$n_c = \frac{m \times n}{t}$

↑ no of non-zero elements in D.

$D = \begin{bmatrix} & & & t_n \\ & & & \\ d_m & & & \end{bmatrix}$

$D = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ $\quad \frac{3 \times 4}{12} = 1$



$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $\quad \frac{4 \times 4}{4} = 4$

b) Heuristic Approach :

Von Rijsbergen

temporal order

tnm

used in new event detection & tracking

1- Process documents (objects) serially (one by one)

2- The first document becomes a cluster by itself.

3- Consider the next document

if it is not similar to existing clusters then it starts its own cluster

else

joins to the most similar cluster (s)

as we change the cluster also change their centroids

Questions:

which similarity measure

similarity treshold

order of processing

20.12 Nisan SIV. 4GB 1.9 Ghz

How to represent clusters (cluster centroids)?

2. * Multi pass Algorithms

A typical approach ← use a seed oriented approach " the heuristic approach.

1- Obtain the initial clusters using an efficient algorithm.

2- repeat

- Generate cluster centroids
- Reassign objects to the clusters according to their similarity to the cluster centroids

until all documents stay in their previous cluster OR

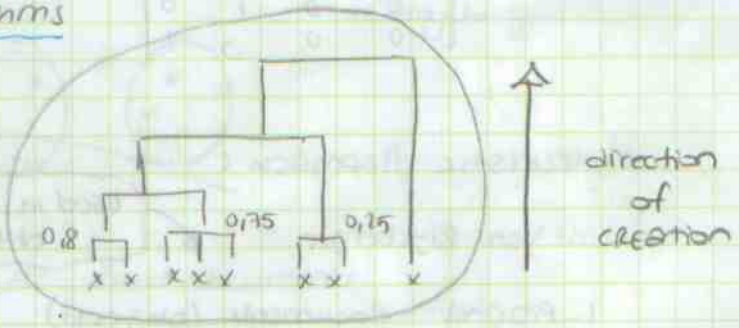
100 % stability
90% "

no of iterations = limit to iterations?

Graph Theoretical Algorithms

Agglomerative

* bottom-up



this cluster is refered as a ⇒ DENDOGREM

Individual similarities are used as a starting point, and a gluing process collects similar items, or group, into larger groups.

Single - link
Complek - link
Average - link

SPSS
SAS
MAth Lab ?

Ellen Voorhees
1985
Salton

Cornell

Single-link: the similarity between a pair of clusters is taken to be the similarity between the most similar pair of documents, one of which appears in each cluster; thus each cluster member will be more similar to at least one member in that same cluster than to any member of another cluster

04/03/2008

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

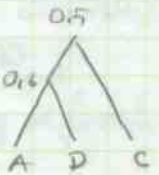
CLUSTER.EXE

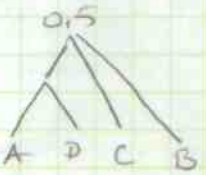
C++

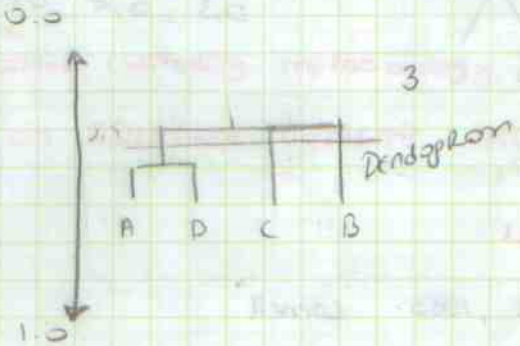
OOL (Object Window Library)

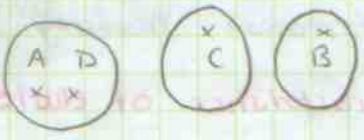
STEP	PAIR	SIMMILARITY VALUE
1	AD	0,6
2	AC	0,5
3	BD	0,5
4	BC	0,4
5	AB	0,3
6	CD	0,3

STEP	Sim Pair
1	AD, 0.6
2	AC, 0.5
3	B,D 0.5









$$\begin{matrix} \\ A \\ \\ \\ \end{matrix} \begin{matrix} \begin{matrix} A & B & C & D \end{matrix} \\ \begin{bmatrix} 1.0 & 0,5 & 0,5 & 0,6 \\ - & 1.0 & 0,5 & 0,5 \\ - & - & 1.0 & 0,5 \\ - & - & - & 1.0 \end{bmatrix} \end{matrix}$$

Simmilarity matrix implied by the dendogram.

S vs Si Product moment correlation

[-1, +1]

K >> total agreement

total disagreement

If the clustering structure reflects the nature of the original data then there will be a high aggreement of the values of S & Si matrices

0,8 → good no for aggreement

↖ Join + ?

Advances in Computes 1985

Complete Algorithm:

the sim between C-D not known thus we can not join this one

① AD 0,6

② AC , 0,5

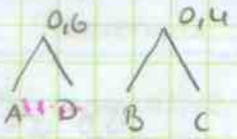
AC , CD ?

③ BD 0,5

AB : ?

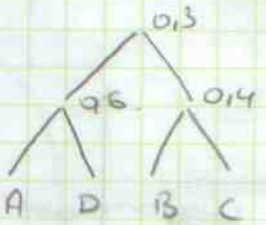
↑↑ singletons the cluster with one member

④ BC 0,4



⑤ AB : 0,3

AB	BD	AC	CD
0,3	0,5	0,5	0,3



Average Link:

Ellen Voorhees, 1985 Cornell

Survey: Peter Willet on hierarchical clustering Information Processing & Management 1988

Desirable Characteristics of Clustering Algorithms:

EFFECTIVE: generates a meaningful clustering structure + provides an effective IR environment

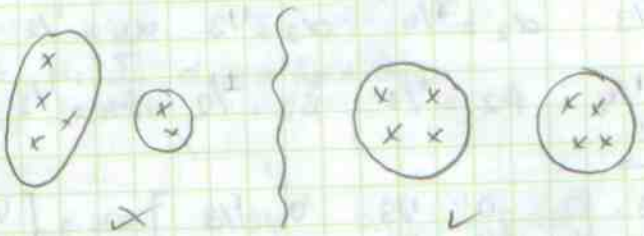
EFFICIENT: Time & Space

1- Order Independence

2- Small errors made during indexing should not effect the clustering structure

3- Clustering structure should be maintainable (notice that usually we work in a dynamic environment

4- Uniform distribution of objects among the clusters.



5- Clustering structure should be stable (addition of new items does not cause huge changes)

6- Small number of parameters (to guess this will make the use of clustering algorithm easier)

Cover Coefficient-based clustering methodology (C3M)

ACM Transactions on Database Systems (ACM TODS)

Dec. 1990

Partitioning
Seed oriented

1- Determine no of clusters (n_c)

2- Find cluster seeds

3- Assign non-seed, to clusters initiated by seed documents

$(m - n_c) * n_c$

non-seeds seeds

m = no of docs
n = no of terms
n_c = no of clusters

Relationship between indexing & clustering:

Example:

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

$\alpha_1 = 1/3 \quad \alpha_2 = 1/4 \quad \alpha_3 = 1/3 \quad \alpha_4 = 1/2 \quad \alpha_5 = 1/2$ → length of 1

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

$$S = \begin{bmatrix} 1/3 & 0 & 0 & 1/3 & 0 & 1/3 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \end{bmatrix}_{5\times6}$$

obtained by using α values

$$S' = \begin{bmatrix} 1/4 & 0 & 0 & 1/2 & 0 & 1/3 \\ 1/4 & 1 & 1/2 & 1/2 & 0 & 0 \\ 1/4 & 0 & 0 & 0 & 1/2 & 1/3 \\ 0 & 0 & 0 & 0 & 1/2 & 1/3 \\ 1/4 & 0 & 1/2 & 0 & 0 & 0 \end{bmatrix}_{5\times6} \quad (\beta)$$

$$S'^T = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 0 & 1/4 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 1/3 & 1/3 & 0 \end{bmatrix}$$

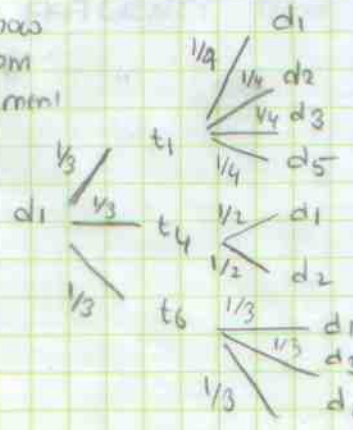
$C = S S'^T$ = result

$5\times6 \quad 6\times5 \quad 5\times5$

$m\times n \quad n\times m \quad m\times m$

$$C = \begin{bmatrix} c_{11} = 1/3 * 1/4 + 0 + 0 + 1/3 * 1/2 + 0 + 1/3 * 1/3 = 0,363 & & \\ & & \\ & & \\ & & \\ & & \end{bmatrix}_{n\times n}$$

We see how to go from one document to another document



$1 \le i, j \le m$

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

↑ Probability of choosing any term of d_i from d_j.

$$c_{11} = \alpha_1 \sum_{k=1}^{6} d_{1k} * \beta_k * d_{1k}$$

$m = 5$, $n = 6$

$$= \alpha_1 \sum_{k=1}^{6} d_{1k}^2 * \beta_k$$

$$= \alpha_1 \left(d_{11}^2 \beta_1 + d_{14}^2 \beta_4 + d_{16}^2 \beta_6 \right)$$

$$= \frac{1}{3} \left(\frac{1}{4} + \frac{1}{2} + \frac{1}{3} \right) = 0{,}361$$

$$c_{12} = \alpha_1 \sum_{k=1}^{6} d_{1k} * \beta_k * d_{2k}$$

$$C = \begin{bmatrix} 0{,}361 & 0.250 & 0{,}194 & 0{,}111 & 0.083 \\ 0{,}188 & 0.563 & 0.065 & 0{,}000 & 0.188 \\ 0.194 & 0{,}083 & 0.361 & 0{,}277 & 0.083 \\ 0{,}167 & 0{,}000 & 0{,}417 & 0.417 & 0.000 \\ 0{,}125 & 0.375 & 0.125 & 0.000 & 0.375 \end{bmatrix}$$

c_{ij} & c_{ji} can be different when they are nonzero

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

2- $c_{ij} > 0 \iff c_{ji} > 0$

3- Row sum = 1

4- c_{ij} and c_{ji} may be non-zero and not equal to each other

↳ c_{12}, c_{21}

0,250 0,188 > 0 $\quad c_{12} \neq c_{21}$

Two documents are identical ⇒ what would happen?

5- $d_i \equiv d_j \Rightarrow c_{ii} = c_{jj} = c_{ij} = c_{ji}$

ABOUT HOMEWORK

(1) Precision

Recall

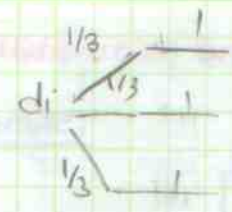
0,0 1,0

TREC 6 Appendix A.

d) find trec eval package

bpref

↳ 11 point representation



$\frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$

If di is unique, i.e, its trms do not appear in any other doc $c_{ii} = 1$

Collection $\Rightarrow$ all docs are unique

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

$$n_c = m = \sum_{i=1}^{m} c_{ii} = \sum_i \delta_i = m$$

$\Rightarrow$ all identical $\begin{bmatrix} \vdots \\ 1/m \\ \vdots \end{bmatrix}$ $\quad n_c = 1 = \sum_{i=1}^{m} \frac{1}{m} = 1$

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

11/03/2008

* C^3M (Cover-Coefficient-Based Clustering Algorithm) Continued ...

Partitioning (flat)

Seed based

n_c : no of clusters is calculated by cc concept.

$C = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$ $\quad n_c = m$ — all docs are unique — $n_c = 1$ — all docs are identical — $C = [1/m]$

$D = \begin{matrix} d_1 \\ d_m \end{matrix} \begin{bmatrix} t_1 & & t_n \\ & \cdots & \\ & & \end{bmatrix}_{m \times n}$ $\quad D = C_{m \times m}$

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

$$D = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \rightarrow \alpha_1 = 1/3$$

$\beta_1 = 1/4$

$$C = \begin{bmatrix} 0,361 & 0,250 & 0,194 & 0,111 & 0,083 \\ 0,188 & 0,563 & 0,065 & 0,000 & 0,186 \\ 0,194 & 0,083 & 0,361 & 0,277 & 0,083 \\ 0,167 & 0,000 & 0,417 & 0,417 & 0,000 \\ 0,125 & 0,375 & 0,125 & 0,000 & 0,375 \end{bmatrix}$$

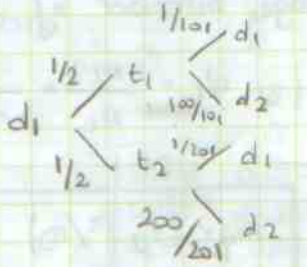
diagonal values gets ↓ as the collection gets ↑

c_{ii} → small no

$c_{ii} \gg c_{ij}$

for a weighted D matrix ⇒ c_{ii} can be < c_{ij}

$$\begin{bmatrix} 1 & 1 & 0 & \dots & 0 \\ 100 & 200 & 0 & 0 & \dots & 0 \end{bmatrix}$$



- D is binary

→ $\min(c_{ii}) = 1/m$

$c_{ii} = c_{jj} = c_{ij} = c_{ji} \Rightarrow d_i \equiv d_j$ ← true for both binary and weighted cases.

$$n_c = \sum_{i=1}^{n} c_{ii} = \sum \delta_i$$

n_c ← decoupling coefficient

$\frac{n_c}{\text{large number}}$

$2,54 * 10^4 \rightarrow 1,2 * 10^{-2}$

$2,54 * 10^4$, 000001.2 ⇒ 0,00000012 $* 10^4$

$\frac{0,00 * 10^4}{2,54 * 10^4}$

GOV2 collection, TREC conference

δ = average decoupling coefficient

$$\delta = \frac{1}{m} \sum \delta_i = \frac{1}{m} \sum c_{ii}$$

n ? m

$$n_c = \sum_{i=1}^{m} \delta_i = m * \delta \rightarrow m * \frac{1}{m} \sum \delta_i$$

Coupling Coefficient $\Psi_i = 1 - \delta_i$

avg. coupling coef $\Rightarrow \frac{1}{m} \sum_{i=1}^{m} \Psi_i$

$$\boxed{n_c = m * \delta}$$

* Average number of docs per cluster

$$d_c = \frac{m}{n_c} = \frac{m}{m * \delta} = \frac{1}{\delta}$$

$$\boxed{\max(1, m/n) \leq d_c \leq m}$$

$$\boxed{\max(1, n/m \leq d_i \leq n}$$

$C = [\quad]_{m \times m}$ n_c

$C' = [\quad]_{n \times n}$ n_c' ← for terms

$n_c = n_c'$

20th March ⇒ form your group & select the project
Thursday

Elements of Style
by Strunk & White

Seed Selection:

Cluster SEED power

highest seed power

docs

[x
x
x
.
.
.
x]

choose top nc docs as the seeds

doc with the lowest seed power

$$P_i = \delta_i * \Psi_i * X_{d_i}$$

δ_i: if its high, provides connection with the collection of docs

X_{d_i}: no of terms in d_i (depth of indexing)

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad 7 \text{ possible}$$

$$C = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

and no connection with the rest of the docs

13/03/2008

	t_1	t_2	t_3	t_4	t_n	t_6
d_1	1	0	0	1	0	1
d_2	1	1	1	1	0	0
d_3	1	0	0	0	1	1
d_4	0	0	0	0	1	1
d_5	1	0	1	0	0	0

$D =$

$$C = \begin{bmatrix} 0,361 & 0,250 & 0,194 & 0,111 & 0,083 \\ 0,188 & 0,563 & 0,065 & 0,063 & 0,141 \\ 0,194 & 0,063 & 0,361 & 0,277 & 0,083 \\ 0,167 & 0,000 & 0,417 & 0,417 & 0 \\ 0,125 & 0,375 & 0,125 & 0 & 0,375 \end{bmatrix}$$

$$P_i = \delta_i * \Psi_i * X_{d_i}$$

seed power

X_{d_i}: no of terms in d_i (α_i^{-1})

$\delta_i = c_{ii}$

$\Psi_i = 1 - \delta_i$

P_i
$d_2 \rightarrow 0,984$
$d_1 \rightarrow 0,692$
$d_3 \rightarrow 0,692$

$P_1 = 0,361 * (1-0,361) * 3 = 0,692$

$P_2 = 0,563 * (1-0,563) * 4 = 0,984$

$P_3 = 0,692$

$P_4 = 0,484$

$P_5 = 0,469$

$n = \sum_{i=1}^{r} c_{ii} \simeq 2$

$c_{11} = c_{33} = c_{13} = c_{31}$?

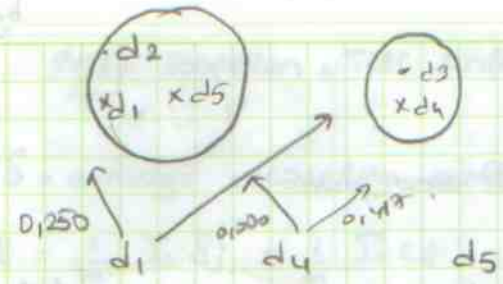
d_1 & d_3 are not the same

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

Choose d_3 since it uses more unused term. The previously selected seed (d_2) contains t_1, t_2, t_3, t_4.

$d_1 \rightarrow t_5, t_6$

$d_3 \rightarrow t_5, t_6$



non-seed
seed
$3 \times 2 = 6$

$(m - n_c)$

* No of computations Involved During Clustering:

$$m + (m - n_c) \times n_c \cong m + m \times n_c$$

↑
needed for finding n_c

$m \gg n_c$

TONS = 214

INSPEC = 12,654

Inverted Index for SEED Documents (IISD)

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

$d_1 \rightarrow$ | t_1 | t_6 | t_{10} | t_{15} |

$d_2 \rightarrow$ | t_{10} | t_{20} | t_{30} | t_{40} |

Sadece d_2, d_3'e bakılıyor.

$t_1 \rightarrow \langle d_2, 1 \rangle \langle d_3, 1 \rangle$

$t_2 \rightarrow (d_2, 1)$

$t_3 \rightarrow (d_2, 1)$

$t_4 \rightarrow (d_2, 1)$ ⇒

$t_5 \rightarrow (d_3, 1)$

$t_6 \rightarrow \langle d_3, 1 \rangle$

cluster d1 :

c_{12}	c_{13}
0	0

t_4

$c_{12} = c_{21} =$

t_3: $c_{12} = c_{12} + \alpha_1 * (d_{11} * \beta_1 * d_{21})$

$= 0 + 1/3 * (1 * 1/4 * 1) = 1/12$

$c_{13} = c_{13} + \alpha_1 * (d_{11} * \beta_1 * d_{31})$

$= 0 + 1/3 * (1 * \frac{1}{4} * 1) = \frac{1}{12}$

t_4

$c_{12} = c_{12} + \alpha_1 (d_{14} * \beta_4 * d_{24}) = \frac{1}{12} + \frac{1}{3} * (1 * \frac{1}{2} * 1)$

$= 0,250$

t_6 $c_{13} = c_{13} * \alpha_1 * (d_{16} * \beta_6 * d_{36}) = \frac{1}{12} + \frac{1}{3} * (1 * \frac{1}{3} * 1)$

$= 0,194$

$\frac{c_{12}}{0,250}$ $\frac{c_{13}}{0,194}$ $c_{12} > c_{13}$ so

d_1 joins the cluster of d_2

Complexity:

Find c_{ii} value $\Rightarrow m * X_d$

X_d = avg no of terms / doc

t_g : term geneardity, avg no of doc / term

Creat IISD $\Rightarrow n_c * X_d$

Clustering

assign non seeds to seeds $\Rightarrow (m - n_c) * \overline{X_d} * t_{gs}$

$\overline{X_d}$ ← no of terms / doc

t_{gs} ← avg posting list in IISD (a term appears in how many different seeds)

$t_{gs} = \frac{2 + 5 \times 1}{6} \simeq 1$

ET $O(\ )$ $O((m - n_c) * X_d * t_{gs}) = O(m * X_d * t_{gs})$

execution time $m X_d t_{gs}$

$m * X_d * t_{gs}$

$\frac{m}{}$ $\frac{m \times n}{t}$

ET

18/03/2007

$O(m \cdot x_d \cdot tg_s)$ → avg posting list IISD

↑ no of docs ← avg no of terms / bc

$M - n_c \cong m$

↑ ↑

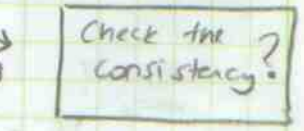
10^6 10^0

CLUSTER VALIDITY:

clusters: meaningful?

D → [C³M] → 

clusters generated by humans



"Text anchored phrase analysis" Kelvin (Yubilak)

Test Collection

1_documents
2_queries
3_relevant docs for queries.

TREC

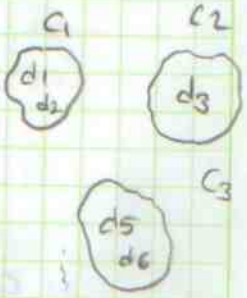
$n_q = 50$

↑ no of queries

Relevant docs

$q_1 \rightarrow 1, 2, 3,$

$q_2 \rightarrow 1, 5, 6$



q_1 → find the target clusters $\{C_1, C_2\}$ $\{C_1, C_3\}$

$q_2 \rightarrow$

In a meaningful clustering environment no. of target clusters for a given query should be small?

Recall "the cluster hypothesis"

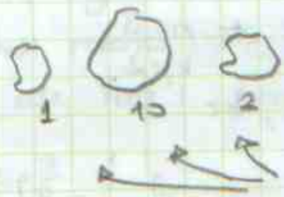
"How small is good enough?"

* Find average number of target clusters for all queries : n_t
(this should be a small number
↳ how small?)

- Keep the clustering structure as the same

n_c : the same
no of documents in C_i $(1 \le c_i \le n_c)$ is kept the same

query'lere hiç bakmadan cluster sayısı her bir clusterin length'i sabit tutulup dökümanları random dağıtıyoruz



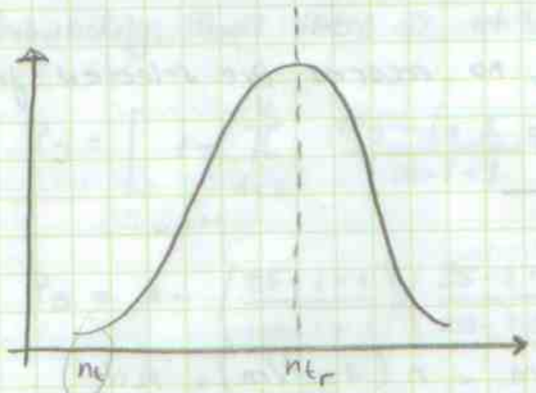
× d_1
⋮
× d_m

1st random n_t
Find n_{tr_1}, n_{tr_2}, ⋮ → take the average → n_{tr}

(random may not be too bad)

$n_t < n_{tr}$

* if $n_t \gg n_{tr}$ ⇒ INVALID CLUSTERING STRUCTURE



Histogram for n_{tr} values

S.B Yao 1977, Communications of the ACM
"Approximating block accesses in database organizations".

Blocks have same size

q → relevant documents
1
2
3
⋮

Alfonso
Cardenas

using his original notations:

(no of terms) ← n : no of records (in each block we have the same no of docs)

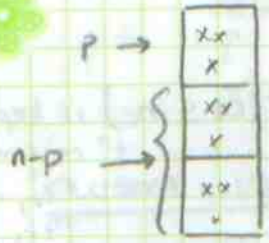
m : no of blocks — the probability of accessing a block

block size = n/m ⇒

k : no of relevant docs for the given query

$p = n/m \Rightarrow$ no of records in the j^{th} block

$n - p$ = no of records in the other blocks



C_k^n = # of combinations that we can have if we try to select k documents at of n documents

a, b, c $\Leftarrow n = 3$, $k = 2$

ab, ac, bc

$$C_k^n = \frac{n!}{k!(n-k)!} \qquad C_2^3 = \frac{3!}{2!(3-2)!} = \frac{3!}{2!} = 3$$

C_k^{n-p} = different values of selecting k documents from $(n-p)$ documents.

$\Rightarrow$ The probability that no records are selected from the j^{th} block

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

Let $d = 1 - 1/m$

$$n - p = n - n/m = n(1 - 1/m) = nd$$

Expected value

$E(I_j)$ = Probability of selecting at least a record from the j^{th} block

$$1 - C_k^{nd} / C_k^n$$

Expected no of blocks to be accessed.

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

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

$$m * \left[1 - \frac{nd!}{k!(nd-k)!} \cdot \frac{k!(n-k)!}{n!} \right] = m * \left[1 - \frac{1 \cdot 2 \cdots nd}{1 \cdot 2 \cdots (nd-k)} \cdot \frac{1 \cdot 2 \cdots (n-k)}{1 \cdot 2 \cdots n} \right] =$$

$$= \frac{(nd-k+1)(nd-k+2)\ \dots\ nd}{(n-k+1)(n-k+2)\ \dots\ n}$$

$$n_{tr} = m * \left[1 - \prod_{i=1}^{k} \frac{nd-i+1}{n-i+1} \right]$$

n_{tr} ⇐ for a query

→ Buradan sonra kendi relation'larımızla devam ediyoruz.

** How to use Yao's formula in a clustering environment?

$k = 3$

$m = 100$ (no of docs)

$|C_1| = 5$ ← no of docs in C_1

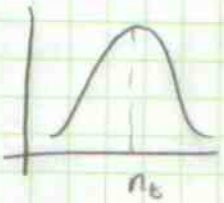
$m_j = 100 - 5 = 95$ → no of docs in the other clusters.

- Probability that there is at least one document in this cluster.

$$P_j = \left[1 - \prod_{i=1}^{k} \frac{m_j - i + 1}{m-i+1} \right]$$

$$P_5 = 1 - \left(\frac{95-1+1}{100-1+1}\right) * \left(\frac{95-2+1}{100-2+1}\right) * \left(\frac{95-3+1}{100-3+1}\right)$$

$$= 1 - \left[\left(\frac{95}{100}\right)\left(\frac{94}{99}\right)\left(\frac{93}{98}\right)\right] \simeq 1 - 0,86 = 0,14$$



→ ~%35 index işlemi (✓) diyeceğiz.

Indexing:

We have two components:

1. Term Selection
2. Assigning weights to terms.

Database Management:

Records ⇒ record attributes stNo, stName, stgpa

* Information Retrieval:

Documents → [Indexing] → $D = \begin{matrix} & t_1 & \dots & t_n \\ d_1 & & & \\ \vdots & & & \\ d_m & & & \end{matrix}$

⇒

What's a term?

Controlled Vocabulary

↑

terms are known beforehand

MESH : Medical Subject Heading
National Institutes of Health

Uncontrolled Vocabulary:

① Stoplist or no stoplist ⇒ use eliminates 30% of terms.

② Stemming vs. no stemming

if stemming then

which algorithm?

[for English → (Porter's Algorithm published in Program)
↳ journal

[for Turkish → agglutinative language
↳ words are created by using suffixes

- simple word truncation

bilgisayar
↳ use first 5 letters (a simple minded approach)
FS

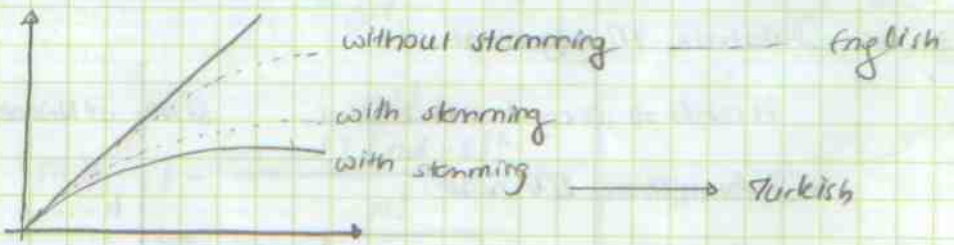
- word morphological structure

actual stem of a word — better → lemma
best → 'good'

its "dictionary entry" → Lemma

CONCERNS OF THE INDEXING PROCESS:

1- what's the best indexing vocabulary



2 - What should be the importance of a term in a document? tf x idf

3 - How many terms in a certain document?

4 - How many documents per term? ← D = [ ]

Gerard Salton AND Chris Buckley (Information Processing and Management)

* 1988, "Term Weighting Approaches in Automatic Text Retrieval" IPM

① D

↑

Query document matching function

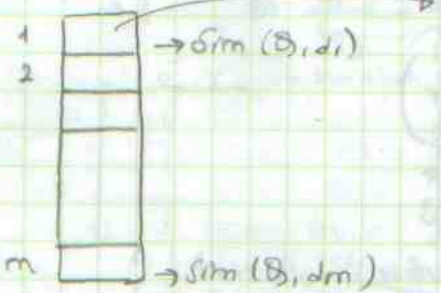
1800 diff

↓

267 unique.

20 Mar. 2008

Simmilarity



$$Sim(Q, d_i) = \sum_{k=1}^{n} w_{qk} \, w_{dk}$$

tf x idf

Term Weighting Components:

Term Frequency Component (TFC)

Collection Frequency Component (CFC)

Normalization Component (NC)

d	q
TFC	TFC
CFC	CFC
NC	NC X → not needed

We'll see that this one is

Unnecessary to normalize query components

TFC

b: binary (1,0)

t: raw term frequency (tf)

n: augmented normalized

$$\text{term frequency} = 0.5 + 0.5 * \frac{tf}{\max tf}$$

max $tf = 5$

document vector (5 0 1 2)

b : (1 0 1 1)

tf : (5 0 1 2)

n : ($0,5 + 0,5 \frac{5}{5}$ 0 $0,5 + 0,5 \cdot \frac{1}{5}$ $0,5 + 0,5 \frac{2}{5}$)

$\frac{\sqrt{-min}}{max-min}$

CFC

X : no change use original TFC component

f : $\ln \frac{m}{tg_i} + 1$

→ NO of document that contain t_j

$tg_i = 3$

P = using a probabilistic inverse collection

frequency factor $\ln \left(\frac{n - tg_j + 1}{tf_j} \right)$

NC : N : no of change $C = \frac{1}{\sqrt{\sum w_i^2}}$

TERM WEIGHT ASSIGNMENT

1800 → 287 diff pos

	document vector		query vector	
3	b, t, n	TFC	TFC	3
3	x, d, p	CFC	CFC	3
2	x, c	NC	NC	1

$(3 \times 3 \times 2) \times (3 \times 3 \times 1)$ => we've this much possibility

Example: max tf

$$D = \begin{bmatrix} 2 & 0 & 1 & 2 & 0 \\ 0 & 2 & 1 & 3 & 1 \\ 2 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} 2 \\ 3 \\ 2 \\ 1 \\ 2 \end{matrix}$$

term grouping

tg: 4 2 2 4 3

$m=5$ $n=5$ 5 docs 5 terms

d_1: tfc → LFC → NC

tfc: term frequency

TFC → t: (2 0 1 2 0)

→ $\int \ln \frac{n}{tg_j} + 1$

4 2 2 4 3

$\ln \frac{5}{4}+1$ $\ln \frac{5}{2}+1$ $\ln \frac{5}{2}+1$ $\ln \frac{5}{4}+1$ $\ln \frac{5}{3}+1$

1.22 1.92 1.92 1.22 1.51

d_1 (2 * 1.22 0 * 1.92 1.92 2 * 1.22 0 * 1.51)

⇒ (2.44 0 1.92 2.44 0)

$$c = \frac{1}{\sqrt{\sum \omega_i^2}} = \frac{1}{\sqrt{(2.44)^2 + 0 + (1.92)^2 + (2.44)^2 + 0}} = \frac{1}{3.95}$$

normalize

$d_1 \Rightarrow \left(\frac{2.44}{3.95} \quad 0 \quad \frac{1.92}{3.95} \quad \frac{2.44}{3.95} \quad 0 \right)$

(0,62 0 0.49 0,62 0)

Query Weight Normalization: nf_x

Q: (1 0 0 2 0)

$tg \Rightarrow$ 4 2 2 4 3 ← coming from D

TFC:n $\Rightarrow$ $(0,5 + 0,5 \cdot \frac{1}{2} \quad 0 \quad 0 \quad 0,5 + 0,5 \frac{2}{2} \quad 0)$

$(0,75 \quad 0 \quad 0 \quad 1 \quad 0)$

CFC $\Rightarrow$ $\frac{n^{\wedge}}{tg_j}$

$(1.22 \quad 1.92 \quad 1.92 \quad 1.22 \quad 1.51)$

$(0,75 * 1.22 \quad 0 \quad 0 \quad 1 * 1.22 \quad 0)$

$\Rightarrow (0,92 \quad 0 \quad 0 \quad 1.22 \quad 0)$

$$D = \begin{bmatrix} 0,62 & 0.00 & 0.49 & 0,62 & 0.00 \\ 0.00 & 0,46 & 0,33 & 0,63 & 0,26 \\ 0,78 & 0,00 & 0,00 & 0,33 & 0,48 \\ 0,63 & 0,00 & 0,00 & 0,00 & 0,78 \\ 0,73 & 0,58 & 1,00 & 0,37 & 0,00 \end{bmatrix}$$

$Sim(Q, d_1) = 0,62 * 0,92 + 0 + 0 + 0,62 * 1.22 + 0) = 1.33$

$Sim(Q, d_2) = 0,77$

$Sim(Q, d_3) < 1.20$ Ranking : $d_1 > d_3 > d_5 > d_2 > d_4$

$Sim(Q, d_4) = 0,58$

$Sim(Q, d_5) = 1.12$

Term Discrimination Value:

We use terms that distinguish documents from each other

Intuition: We will be able to identify the relevant docs.

↑ effective

Document Space

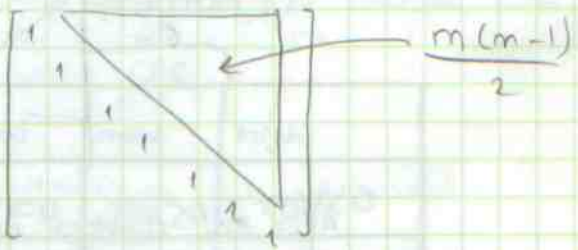
using terms with better discrimination power

using terms with low discrimination power

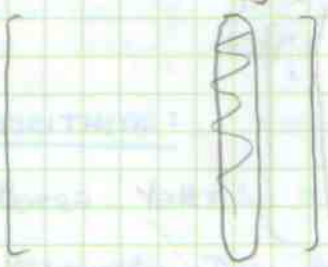
Q_i: space density

$Q = \sum_{i=1}^{m} \sum_{j=1}^{m} I_{ij} \quad S_{ij} \quad i \neq j$

$m(m-1)/2$



t_j ← delete this term

$D =$ 

Q_{-j}: space density without t_j

t_j is a good discriminator ← (what's its effect on Q (space density))

using all terms → using all terms except t_j

$Q < Q_{-j}$

all terms θ_j | without t_j $\theta_{\bar{j}}$

term characteristics

Good discriminator $<$

Bad discriminator $>$

Indifferent discriminator $\approx$

$TDV_j = \theta_{\bar{j}} - \theta_j$ $\quad TDV = \frac{\theta_{\bar{j}}}{\theta_j}$

Theory of Indexing Salton

$d_{ij} \times TDV_j$

How to calculate TDV using the cover coeff. concept

$$C = \begin{bmatrix} & & \\ & C_{ii} & \\ & & \end{bmatrix} \qquad n_c = \sum_{i=1}^{m} C_{ii} = \sum_{i=1}^{m} \delta_i$$

n_c = # of clusters using all terms

$n_{c\ell}$ = # of clusters without t_ℓ

Good discriminator $n_c > n_{c\ell}$

$$D = \begin{bmatrix} & t_\ell & \\ & | & \\ & | & \\ & | & \end{bmatrix}$$

1- Straightforward approach: requires many computations

Consider θ_i: space density approach

Generate a collection ~~cluster~~ centroid

	t_1	t_2	t_3	t_4	$C_{cent.}$
	1	0	2	1	
	0	0	1	2	
	3	1	0	1	

$C_{cent} = [4/3 \quad 1/3 \quad 3/3 \quad 4/3]$

$n = m \times n$ $\quad t$ $\quad n_{t\ell}$

$$Q = \sum_{i=1}^{m} \frac{sim(d_i, C_{cent})}{m}$$

$$n_c = \frac{m \times n}{t} \qquad n_{c_L} = \frac{m \times (n-1)}{(t - t_{g_L})}$$

27 Mar. 2008

JASIS: (T)

Journal of the American Society for Information Science

Q	n_c
t1	t1
t5	t5
t4	t3
t3	t4
|	|
|	|
|	|

→ show the consistency of rankings
Wilcoxon (Spelling?)
(Spearman's)

TODS 214

INSPEC
(12,684 14,...)
m n

	low	CFC med	high
Q	TDV ≈ 0	TDV > 0	TDV < 0
n_c	TDV > 0	TDV ≈ 0	TDV < 0

↑ TDV's

* STEMMING ALGORITHMS:

Frakes & Baeza - Yates

Information Retrieval: Data Structures & Algorithms Chp 8

wanted → want
wants → want

(stem) ← find approximate root.
root

lemma ← find the dictionary entry for a word

better → good
best → good

word lemma

* English

Porter's stemming Algorithm

Program

* Turkish

agglutinative language

root, suffix, suffix

F5

first 5 characters

simple truncation

milliyet

F5

F6

F3 first 3

F4

F5

F8 first 8

Best in terms of info. ret

↳ how this is decided

base line : no stemming

↓

by comparing the performance with the base line ≈ %30 increase in performance

m ≅ 400.000 docs

5 years milliyet

Stemmer vs. Lemmatizer

F5

Zemberek

NLP tools for Turkish

• Prb about Lemmatizer:

⇒ it gives you a number of choice.

random

(how to chose the best one?)

* How to chose one of the possible lemmas?

- Kemal Oflazer

* disambiguated text

word → adj

word noun

word pronoun

word type

Most frequent word type

noun
verb
adjective
adverb ..?

Turkish

less frequent word type

Chose the one which is most frequent in Turkish?

next verb

⇓

Performance was better

⇒

Turkish

(...) no stemming
F5
lemmatizer
Successor Variety
} 30% higher

- to measure the performance:
  - bprf
  - tree-eval package

Stemming (Conflation) Methods:

- Manual
- Automatic
  - Affix removal
    - removal (prefix, suffix)
  - Successor variety
  - Table look-up
  - N-Gram

(Successor Variety:
Root → after real root → many different letters
suffix → they all begin with different letters
how many different letters)

AFFIX REMOVAL:
removes suffixes & prefixes or both leaving a stem

SUCCESSOR VARIETY:

uses the frequencies of letter sequences in a body of text

N-GRAM:
makes decision based on the number of digrams or n-gram they (words) share

n-gram approach

bigram
trigram
ngram

bigram
statistics ⇒ st, ta, at, ti, is, st, ti, ic, cs

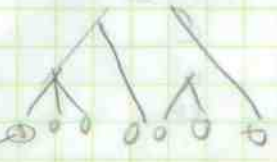
statistical ⇒ st, ta, at, ti, is, st, ti, ic, ca, al

word1 word2 ... wordn

word1
word2
:
wordn

← find the similarity among words by using n-grams

use a clustering algorithm

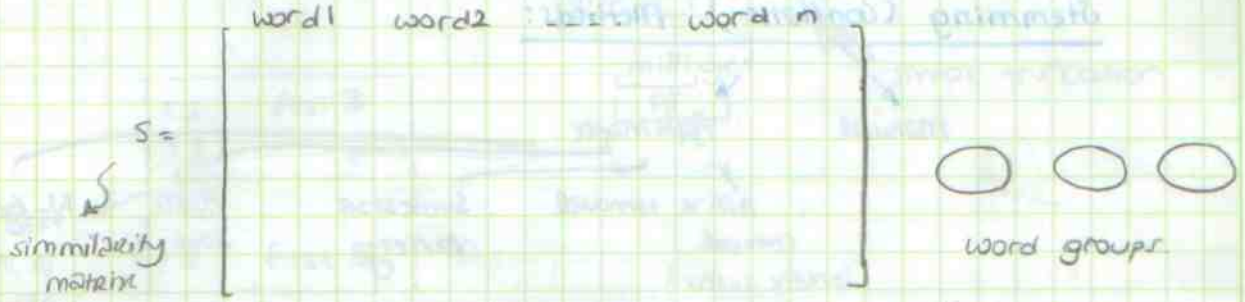


words in the same group are assumed to have the same stem

1st April 2005
(alternatively 2106 01 00)

...Stemming Algorithms:

Affix Removal
n-gram
table backup
Successor variety



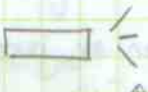
statistical
statistics →
2 gram
st, ta, at, ti, is, st, ti, ic, cs

	st
word1 →	1
word2 →	2

Successor Variety (SV)

SV tries to determine word stems

consider the prefixes of the word.



↑ determine the different letters in all words with the same prefix.

3-prefix
first three
chudas

CORPUS : ABLE, APE, BEATABLE, FIXABLE, READ, READABLE, READING, READS, RED, ROPE, RIPE

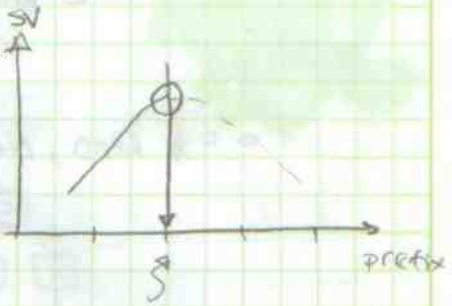
WORD : READABLE

READABLE

PREFIX	SUCCESSOR VARIETY	LETTERS	
R	3	E, I, O	READABLE, READING, READS, RED, ROPE, RIPE
RE	2	A, D	
REA	1	D	

⇒D

PREFIX	SUCCESSOR VARIETY	LETTERS
READ	3	A, I, S
READA	1	B
READAB	1	L
READABL	1	E
READABLE		



prefix is the stem

Information Storage & Retrieval

W... , B...

- no stemming
- lemmatizer_based stemmer
- successor variety
- first n characters (F5, F6, ...)
  - ↳ prec @ 10 || 20 F6 best

matching function

MF1 , ... , MF8

stemmers

- no stemmers
- SV
- F6

we've 24 different combinations

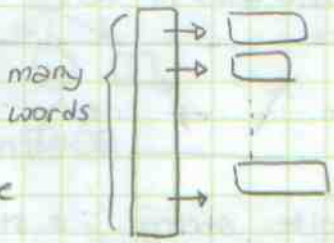
$8 \times 3 = 24$

2400 ≈ 600

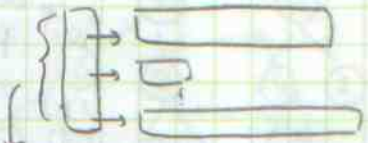
* Why stemming?

- effectiveness
- efficiency

↳ inverted index structure



without stemming



less no of words

Stemming:

Problems

Over stemming!

□ ← (short stems)

understemming

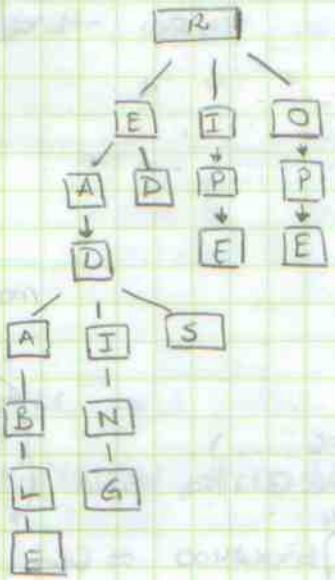
▭ → (very long stem)

How to Store Words?

Knuth Sorting & Searching Tree Structure

Symbol Tree Ex:

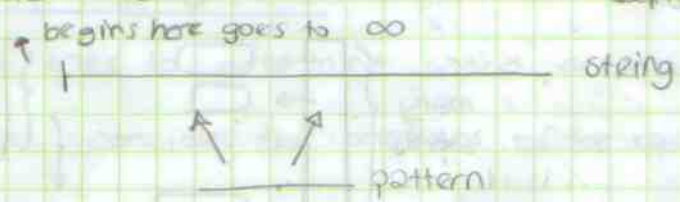
* READ, READABLE, READING, READS, RED, ROPE, RIPE



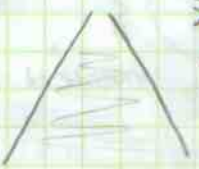
PAT TREE CONSTRUCTION:

Patricia Tree

alphabet = {0,1}



Semi infinite string : s-string (sistring)



SYMBOLS

□: leaves

○: internal nodes

Problem:

Given the text: 0 1 1 0 0 1 0 0 0 1 0 ...

Show the tree Patricia tree after the first eight sistring are inserted.

Bit position:	1	2	3	4	5	6	7	8	9	10	11
	0	1	1	0	0	1	0	0	0	1	0

sistring:

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

2: 1 1 0 0 1 0 0 ...

3: 1 0 0 1 0 0 ...

4: 0 0 1 0 0 ...

5: 0 1 0 0 0 1 ...

6: 1 0 0 0 1 0 ...

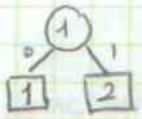
7: 0 0 0 1 ...

8: 0 0 1 0 ...

Insert sistring one by one:

Insert 1: (i.e sistring 1) [0 1 1 0 0

Insert 2:

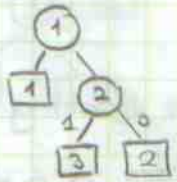


in order to differentiate 1st sistring from 2nd

1 → 0 1 1

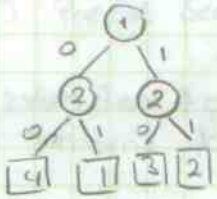
2 → 1 1

Insert 3: 1 0 0 1

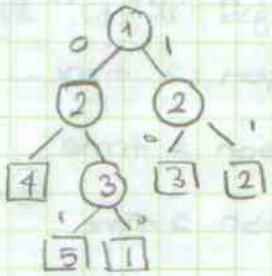


Insert 4:

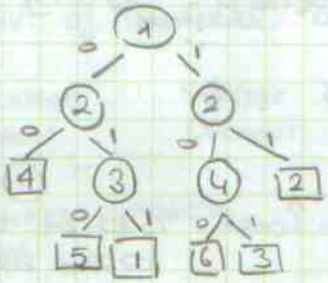
0 0 1 0 0 0



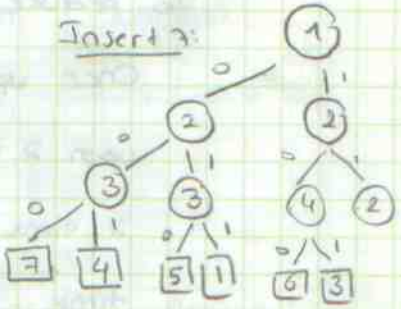
Insert 5: 0 1 0 0 0 1



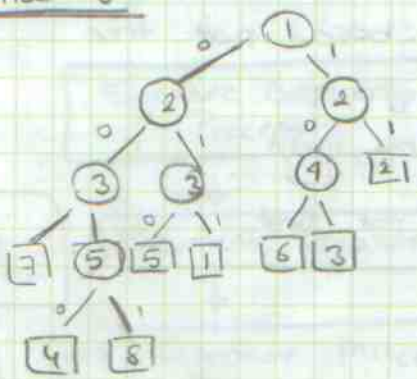
Insert 6:



Insert 7:



Insert 8:



What kind of characteristics do we have in the node of the tree structure?

7 → 0 0 0 1 . . .

4 → 0 0 1 0 0 . .

8 → 0 0 1 0 1

5 → 0 1 0 0 0

1 → 0 1 1 0 0 1 0

6 → 1 0 0 0 1

3 → 1 0 0 1 0

2 → 1 1 0 0 1

increasing order

Write these positions in an array?

| 7 | 4 | 8 | 5 | 1 | 6 | 3 | 2 |

← PAT Array (PAT)

$7 < 4 < 8 < 5 < 1 < 6 < 3 < 2$ in lexicographic order

We're looking for an order

Take a number search it in the array

('binary search')

08/04/08

Another sistring Example

String : Once upon a time, in a far away ..

Sistring1: Once upon a time..

sistring2: nce upon a time..

sistring3: ce upon a time...

no of leaf nodes = n
no of internal nodes = n-1

To reduce the cost generate sistring by only using words

Once upon a time ..

upon a time

a time ← ↑cost: %20 of the character based approach

time

String: an exampl for a word-based pat tree

1	2	3	4	5	6	7
that	is	that	,	that	is	not	that

sistring1: that is that , that is not that

sistring2: is that that is not

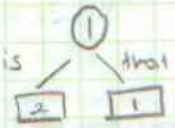
sistring3: that that is not

sistring4: that is not

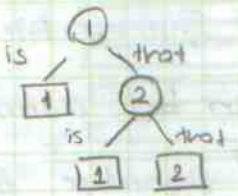
sistring5: is not

Insert sis1,

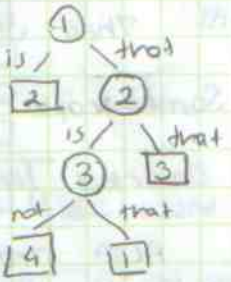
Insert sis2,



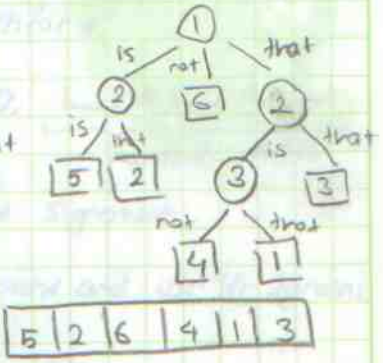
Insert sis3;



Insert sis4



Insert sis5



5	2	6	4	1	3

Chap 5: William Frakes, & Ricardo Baeza Yates

Info Ret: Data & File Structures

use of pat tree → Prefix Search: Find the words beginning with the same prefix.

pat 1 ↔ pat 2
in words

※ Signature Files:

1980s Univ. of Montreal

Ph.D Supervisor : Stavros Christodulakis
Ph.D Student : Christos Faloutsos

bit map ↔ Gotlieb
concept

Document

↓

Signature Generation Process

↓

Document Signature

Queries

↓

Query Signature Generation

↓

Query Signature → Signature matching → False Drop Resolution → TRUE MATCHES

True matches
False matches

not like inverted files
does not have exact representation
because of hashing
(it may be by choice) i.e the document will not match the query but its signature will

false match ≡ false drop

true match ≡ true drop

Why signature files?

Their size < Inverted Files

Some people question this claim?

Inverted files vs. Signature files

ACM Trans. on Database Systems
John Faber

Signature Generation Methods

- Word Signatures
- Superimposed Signatures
  - OR

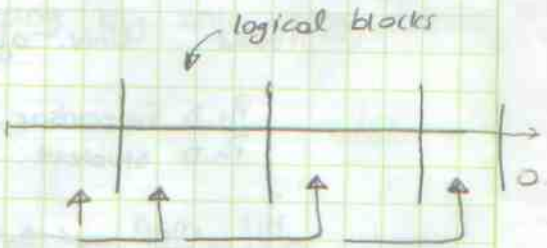
* Document Signature Generation Process:

1- Find term signatures

2- Superimpose (OR) term signatures to obtain document signature

EXAMPLE	TERM	TERM SIGNATURE
object	1000	1000
signature	0010	0100
generation	1000	0100
	1010	1100

block signature D_s



Each block contains the same no of distinct words (the last block may have fewer) no of words

QUERY	QUERY SIGNATURE	RESULT
database	1100 0000	no match
generation	1000 0100	true match
information	1010 0000	false match

Query-signature = Q_s
document-signature = D_s

If Q_s and D_s = Q_s then

We have a true/false match

else

D does not match query

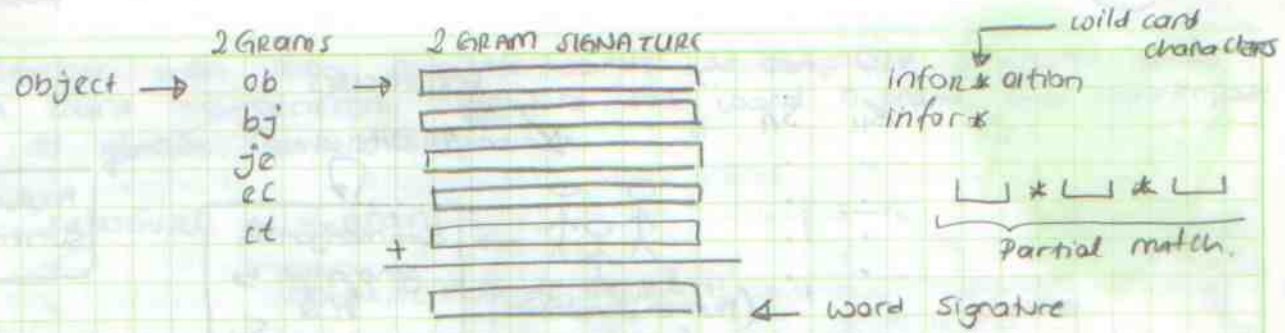
since in our query there is 1 word
here query signature = term signature

D has a match but false (Does it really contain the query term? → No!)

term: information:
1010 0000 & 1010 1100 = 1010 0000 match
but query doesn't contain the term (false match)

Prob of having a false match is very low

False Drop Resolution → to see whether it is a false or true match.



Do not use the word itself, but generate '2grams' for the word and use its 2grams

Construction of Word signatures (without using the n-gram concept)

n: signature length (F)

w: no of 1s in a term signature (weight)

k: no of 1s that we want to have in a signature

Let's say we want to have;

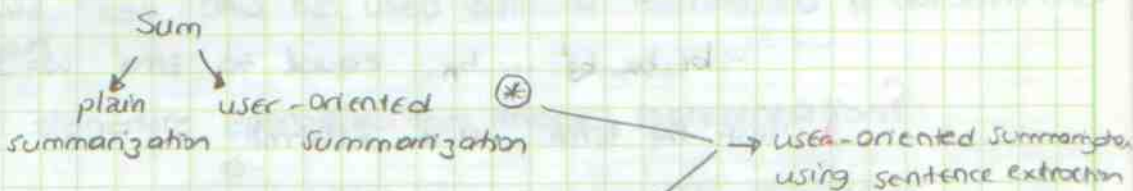
Sallon: (his blue book)
Text

- signatures with the same weight

1- Initialize $b_1\ b_2\ b_3 \dots b_n$, $w=0$
$0\ 0\ 0 \dots 0$

10/04/2008

User-Oriented Summarization:



SUMMARIZATION TYPES

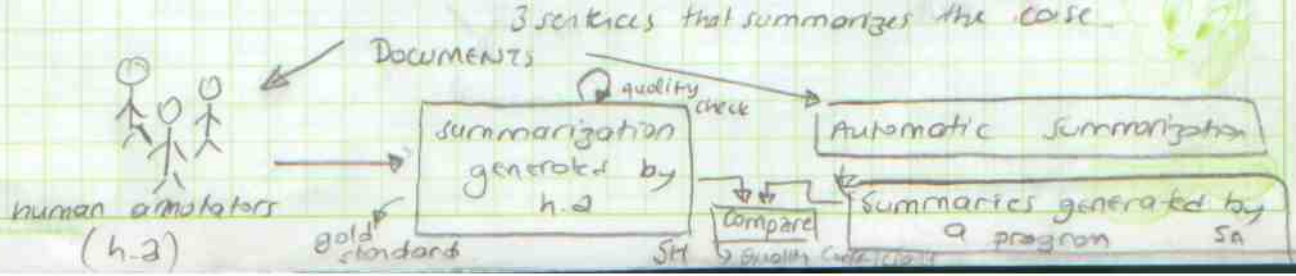
1- based on sentence extraction

2- abstraction : construct a summary without using the original sentences.

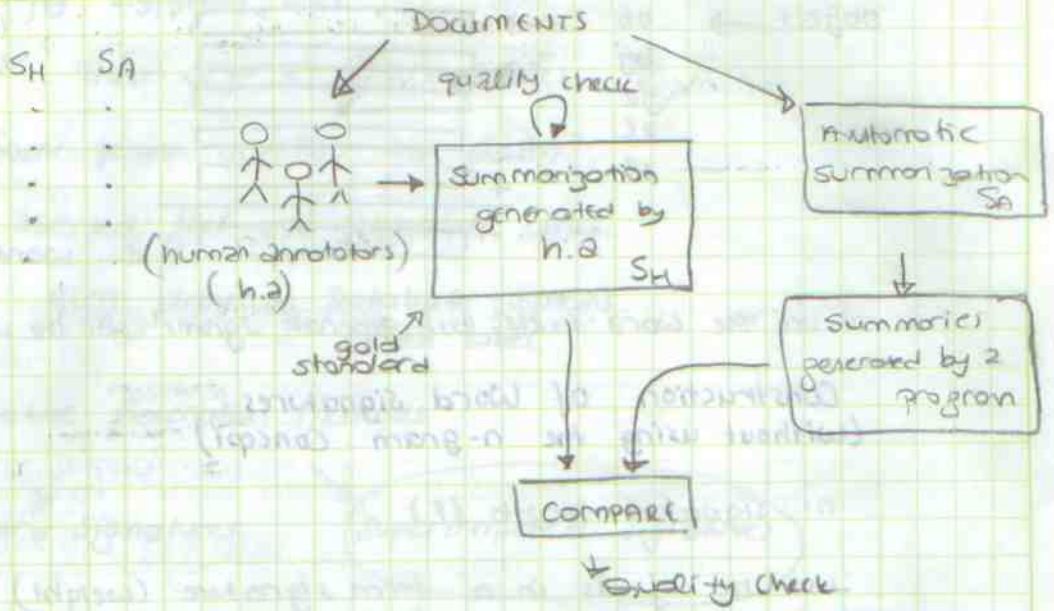
DUC : Document Understanding Conference

TAC : Text Analysis Conference

CNN.com → display news
meanwhile
3 sentences that summarizes the case



139.179.21.48 / Ross / login.aspx CS user CS pass



ooo Signature Files (cont'd)

Term Signature Generation:

From Salton: ... TEXT ... (fig 7.42)

n: Signature length ← F
w: no of 1's in a term signature (so far)
k: no of 1's we want to have in a signature ← S

$F \gg S$ $n \gg k$

1. Initialize

$b_1\ b_2\ b_3 \dots b_n$ equal to zero $w = 0$

2. Hash the term $H \leftarrow h(\text{term})$

3. Initialize the random generator using H.

4. Determine next vector position to the be set equal to 1

$j \leftarrow \lfloor n.\text{random}() \rfloor + 1$

5. Is vector position = 1?
if $b_j = 1$ then goto step 4

6. $b_j \leftarrow 1$; $w = w + 1$

$0 < \text{random}() < 1$

7. If $w < k$ then go to step 4

8. exit.

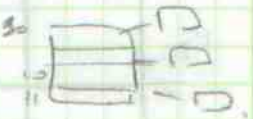
Rather than using complete words we can use n grams of a given word and generate signature for word n-grams and superimpose them to obtain the term signature.

retrieval ← 2-grams

re, et, tr, ri, ie, ev, va, al

So that we can later search terms that involve wild card characters

Compute * ⇒ computer
computation
computational



Extendable Hashing

Use of Signature Files in Different Applications:

Paul Zezula, et al
Dynamic Partitioning of Signature Files
ACM. Trans on Information Systems (ACM TOIS)

Oct 1991, pp 356-367 (LHSS)

← Linear Hashing for superimposed Signature
Quick Filter

1- Text Retrieval
2- Multimedia Representation and Retrieval ← unformatted data
3. Database Operations ← *A. Tharp ← formatted data
4. In prolog database to store and access knowledge
5. CAD: Computer Aided Design.
6. OODB indexing SIGMOD

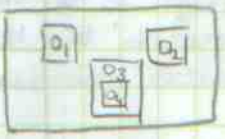
≡

! SIGNATURE FILES CAN BE USED BOTH IN FORMATTED & UNFORMATTED ENVIRONMENTS

How to use signature approach for image representation?

Image (I) → [Image Analysis Process] → Description of image in terms of the object contained

$I = O_1, O_2(O_4, O_5(O_6)), O_3(O_9, O_{10}), O_1, O_3(O_9, O_6, O_7)$



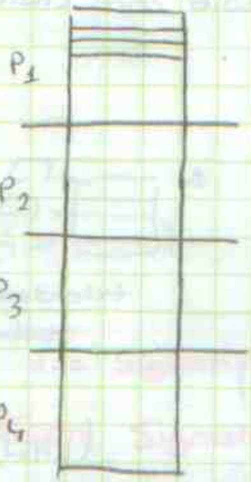
Object ⇒ Text Image

Signature ⇒ Stext Simage

Sobject

Signature File Structures:

- Single Level
  - Sequential Signatures
  - Vertical Partitioning
    - Bit Sliced
    - Frame Sliced
- MultiLevel
  - Signature Tree (A. Thorp)
  - Two Level
- Horizontal Partitioning
  - LHSS
  - KeyBased
  - Gustoffon's method



IEEE TORDE
Transaction on knowledge & Data Eng
Lin, Faloutsos

Aktuğ, Can
Survey
in word doc
my notes

- Sequential Signatures (SS):

Other Names: single Level signatures, bit_string approach

a bit may be 0 || 1 with equal chances

S1	0001	1110
S2	1100	0001
S3	0011	1100
S4	1100	0011
S5	0011	0110
S6	1100	1001

[Add new comers to the end of the file.

Simple Maintanance

Problem Search Process, $O(n)$

17 / 04 / 2006

SSF (Sequential Signature Files)

S1	0001	1110
S2	1100	0001
S_3	0011	1100
S_4	1100	0011
S_5	0011	0110
S6	1100	1001
⋮		
Sn		

←F→

F: signature size
n: no of signature

File size (bit) = $n * F$

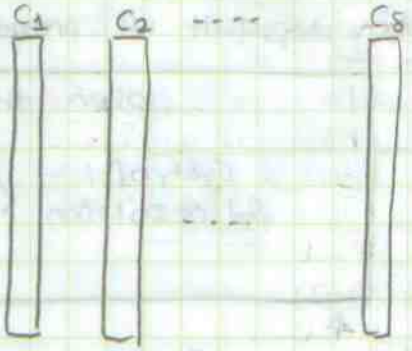
if (Q_n and D_1 = Q_1)
then access the document and perform false drop resolution
↑ make sure that doc contains the query terms

$Q = 1001\ 0000$

Bit Sliced Signature files:

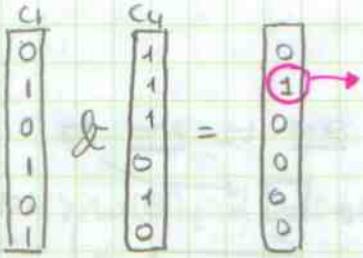
Vertical Partitioning

C_1 :	010	101
C_2 :	010	101
C_3 :	001	010
C_4 :	111	010
C_5 :	101	001
C_6 :	101	010
C_7 :	100	110
C_8 :	010	101



Consider the same query:

Q_5 : 1001 0000



Just consider Doc 2 for false drop resolution

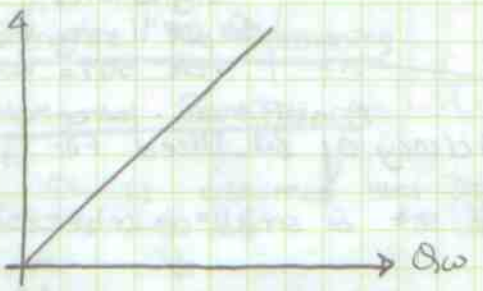
How to store them

Pages in disk

C_1	C_2	C_3	...

Cost of query processing is proportional to Q_w

(no of bits set to 1 in the Q_5).



Q_w : query weight: no of 1s in the query resolution.

Partial query matching can also be used in inverted files.

If we assume that bit density 50%
After processing the first bit of the query we eliminate 50% of the documents (only 50% we remain active)

After processing i no. of query bits only $\left(\frac{1}{2}\right)^i$ no of documents remain active

After a certain point rather than doing bit processing we may prefer to perform false drop resolution since cost of false drop resolution can be lower than bit processing
(Assuming that we have very tall bit slices.)

Not using all of the bits of the query signature is referred to as partial evaluation.

When to quit bit processing?

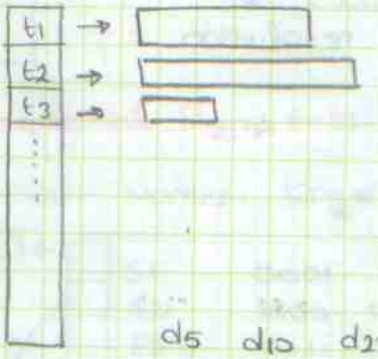
When the

Cost of fd resolution < Cost of bit slice processing + start fd resolution

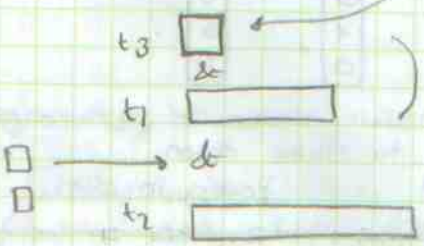
Information Processing Letters, Kocberber, Can 1996
Partial Query matching

Partial Query matching: can be used in inverted files

Inverted Files:



Q: t1 and t2 and t3



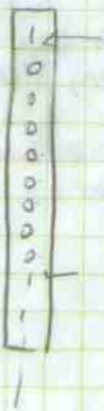
take intersection of these two posting lists

How about using very large signatures?
So we'll may be able to eliminate not 50% but ~90%

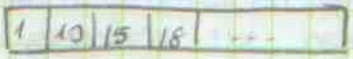
How To Increase The Efficiency of Bit_Sliced File Query Processing

Use long signatures & set a small number of bit locations equal to 1

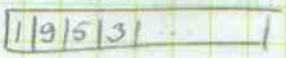
90% = 0 bits
10% = 1 bits



Keep the same information like a posting list



Use dgap approach

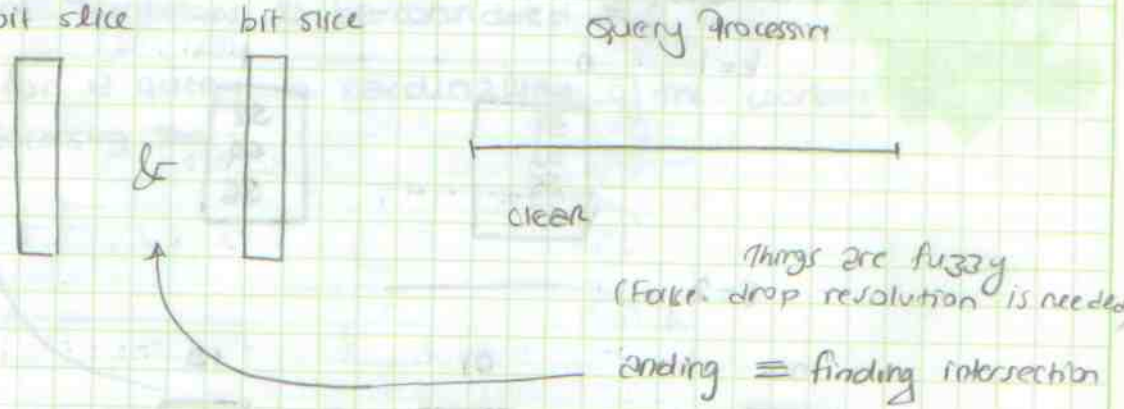


We have smaller numbers

query processing exactly the same as inverted file approach

bit-slice & inverted her ne kadar farklı görünse de, bir yerden sonra birbirine benzemek

If we only store the locations of the 1 bits in a bit-slice then the structure becomes very similar to a posting list



If i) $w(Q)_t$ = no. of bits in a t-term query
↓
# of bits in a signature

then $i = w(Q)_t$
i.e the volume of i cannot be greater than the # of bits in the query.

→ Divide the signature file into parts. In order to ↑ the efficiency of the query processing, in some way we will ignore some partitions.

Signature File Partitioning

Horizontal Partitioning

It is assumed that signatures are random_bit vectors
(50% - 50%)
↓ ↓
1 0

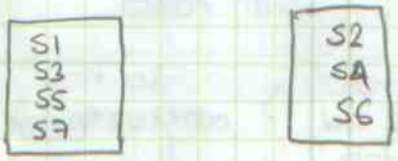
We have following set of signatures:

S1	0111	1000
S2	1000	1011
S3	0011	1100
S4	1100	0011
S5	0110	1100
S6	1001	0011
S7	0000	1111

Fixed Prefix Partitioning: (FPP)

- Just consider 1st bit of the signature

k=1

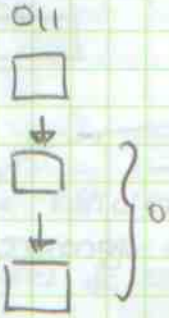


k=2

k=3

PROBLEM: We choose k value
We allocate a page for one block
We wait uniform distribution among partitions since %50-%50

If one of the blocks overflows we will have overflow pages ← Problem



Q1 : 1000 0111
Q2 : 1111 0000
Q3 : 1100 0110

Working sets of query : a set of partitions such that (for a given query)

$\{P_i \mid P_{key} \cap Q_k = Q_k\}$

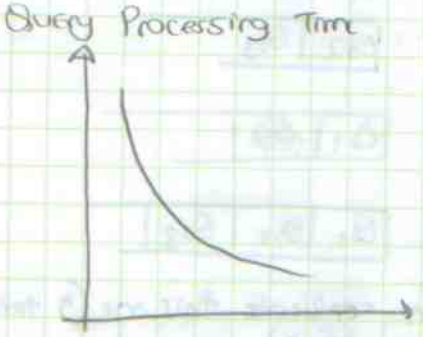
Query	k=1	k=2	k=3
Q1 1000	1 (1)	2 (10, 11)	4 (100, 101, 110, 111)
Q2 111..	1 (1)	2 (11)	_ (111)
Q3 1100	1 (1)	2 (11)	2 (110, 111)

↗ # of partitions (matching partitions)

Can we estimate the # of partitions to be considered?

# of partitions to be considered = $2^{\#\text{ of 0's in k-prefix of}}$

for a query = cardinality of the working set



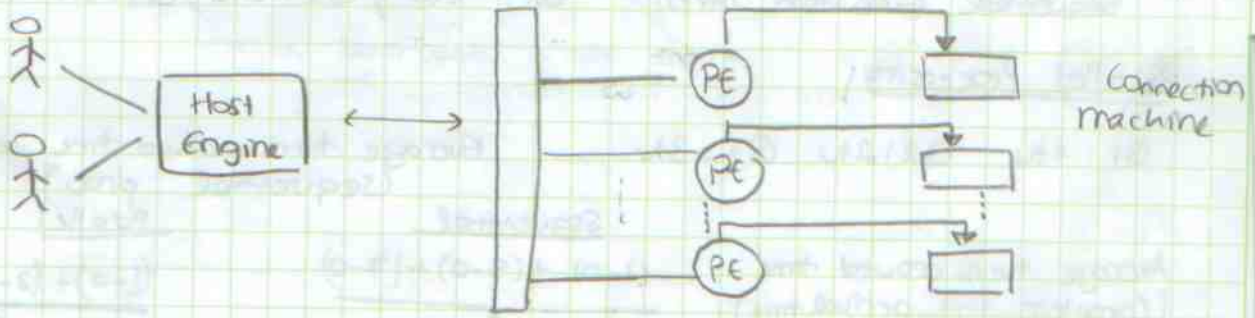
The partition that has all 1's is always selected
So to decrease the processing time, we would like to always keep this partition in the main memory.

- The partition with P_k = all 1s ($k=1 \rightarrow 1$, $k=2 \rightarrow 11$, $k=3 \rightarrow 111$) is always in the working set of the query.

Partition Processing can be done in - Sequential OR
Parallel Partitioning.

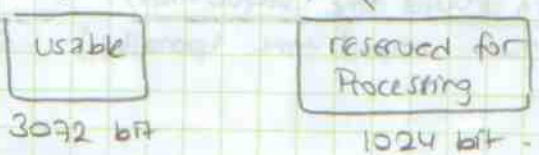
Parallel Hardware Structure:

PE : Processing Element (CPU)



Connection Machine

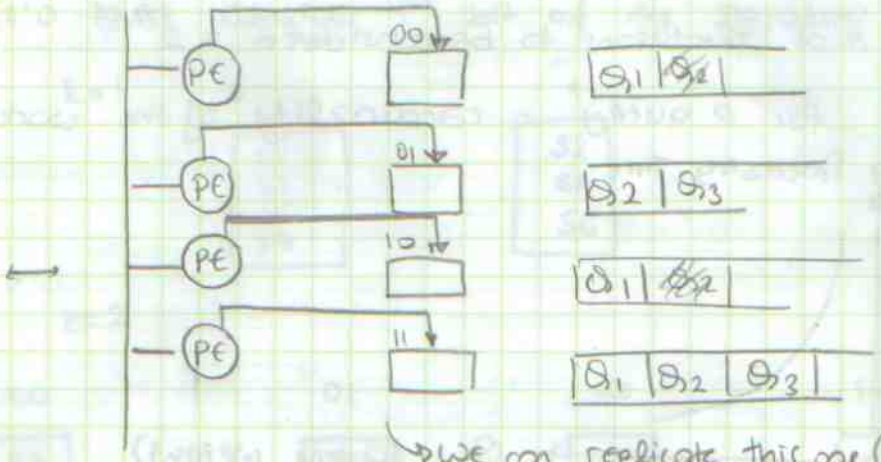
Each PE has a memory of 4056 bits



3072 bit

1024 bit.

Assume we have 4 partitions
we want to compare sequential & parallel processing.



ASSUMPTION: All queries arrive at the same time (t=0)
We will place queries in the queue

Another set of queries:

			Query	k=2
Q1:	1110	0001	Q1	1 (11)
Q2:	0000	1111	Q2	4 (00, 01, 10, 11)
Q3:	0110	0011	Q3	2 (01, 11)

Sequential Processing:

tu: time unit

Q1	Q2	Q2	Q2	Q2	Q3	Q3

t=0 1 2 3 4 5 6 7

Sequential Execution Times: Q1: 1tu ; Q2: 5tu ; Q3: 7tu

Parallel Processing:

Q1: 1tu Q2: 2tu Q3 = 3tu.

Average turn around time (sequential proc.)

Average turn-around time (Complete time - arrival time)

Sequential: $\frac{(1-0)+(5-0)+(7-0)}{3}$ Parallel: $\frac{(1-0)+(2-0)+(3-0)}{3}$

$13/3 = 4,33$ tu $6/3 = 2$ tu.

Throughput: # of jobs completed per unit time: $3/7$ $3/3 = 1$.

Speed Up Ratio: $\frac{\text{Avg. Turn around time (Sequential)}}{\text{Avg Turn around time (parallel)}} = \frac{13/3}{6/3} = \frac{13}{6} = 2,166$

# of p

No of partitions to be accessed for a query = no of 0's in query key

Q = 1000 1101

Problem: # of partitions is fixed, if overflow occurs it has to be recognized.

** SIGNATURE FILE FOR STRUCTURED RECORDS **

Alon Thorp File Organization & Processing p 152 →

Employee (EmpNo, DeptName, Salary, No of Dependents)

Query Types:

1- based on primary attribute

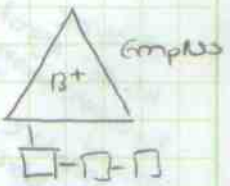
Display EmpName where EmpNo=100

2- based on secondary attributes

Display EmpName where Salary=2000

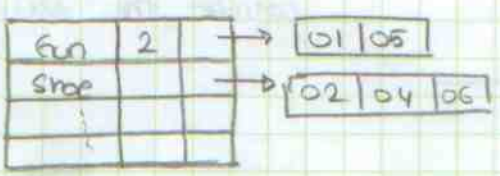
** Signature Files vs. Inverted Indexes for DB systems: **

EmpNo	EmpName	Dept	NoOfDependent	Salary
01	Jane	Gun	1	
02	George	Shoe	2	
04	Dave	Shoe	3	
05	Mary	Gun	2	
06	Mike	Shoe	1	

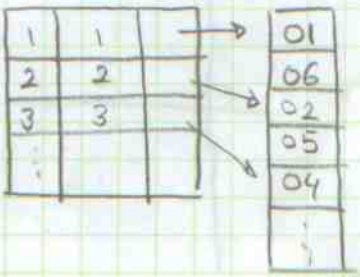


Inverted Indexes for the secondary Attributes:

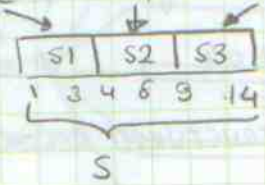
Inverted Index for Dept



Inverted Index for NoOfDependents



EmpName Dept Salary — use Hashing to set the signature bits



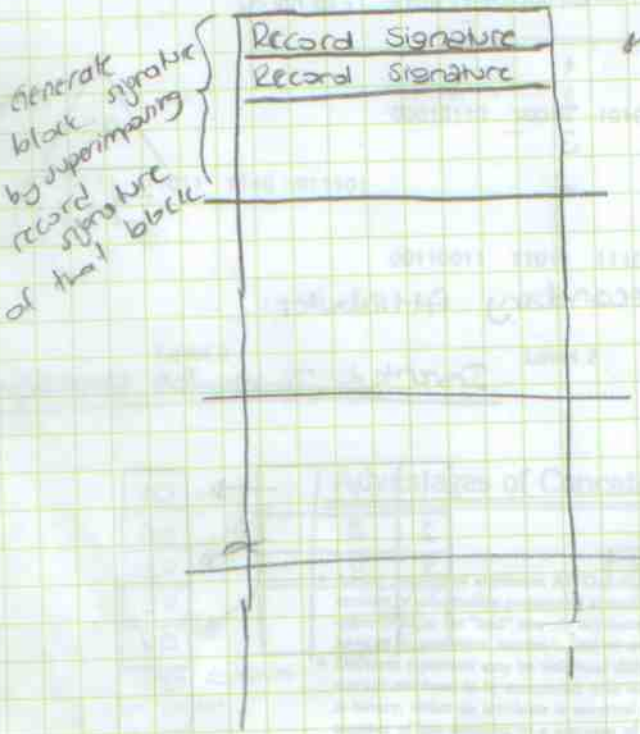
← record signature

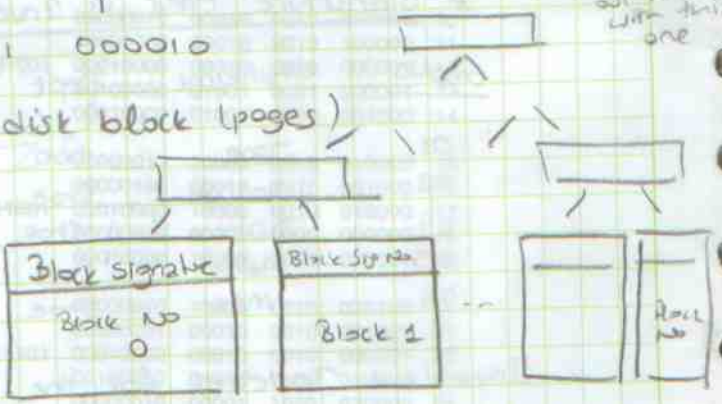
EmpName	s(1) = 1	if EmpName begins with	[A, E]
	s(2) = 1	if " " "	[F, L]
	s(3) = 1	if " " "	[M, Z]
	s(4) = 1	if DeptName	[A, E]
	s(5) = 1	if	[F, J]
	s(6) = 1	if	[K, M]
	s(7) = 1	if	[N, R]
	s(8) = 1	if	[S, Z]
	s(9) = 1	if EmpSalary ≤ 10,000	
	s(10) = 1	if EmpSalary	[10 000, 20 000]
	s(11) = 1		[20 000, 25 000]
	s(12) = 1		[25 000, 30 000]
	s(13) = 1		[30 000, 45 000]
	s(14) = 1		≥ 45 000

EXAMPLE

123 John Doe Toy 35 000

010 00001 000010





First match query signature with super block signature if it matches the block signature consider the next level. . .

28 April 2005

How many bits to be set for a term?

to produce signatures with

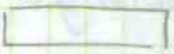
50% 0 and
50% 1

SIGMOD Conf? Faloutes

$$S_{opt}(m_{opt}) = \frac{F \ln 2}{D} \simeq \frac{0.7F}{D}$$

F: signature size in bits
D: avg no of terms logical block

D=3



Another method by Faloutus & Christodoulakis

classifies terms according to their discrimination power (or more important terms ≡ terms which are more frequently used in queries) are allowed to set more no of bits.

↗ decrease % of false matches

How can we make sure that we have the same no of 0s and 1s in a block signature?

Leng & Lee 1992 Acm Tods ← Trans on Database Systems

1. $S_1\ S_2 \ldots S_I = 0$ (initialization)
2. while $s_w < (F/2)$ {

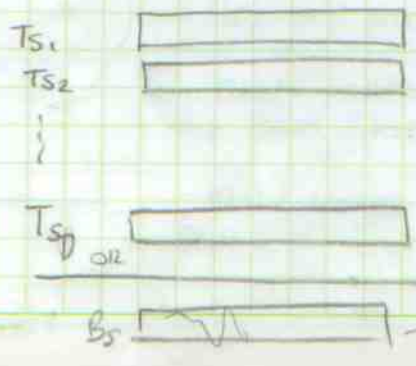
take the next term & generate T_S (term signature)
$S = S$ or T_S

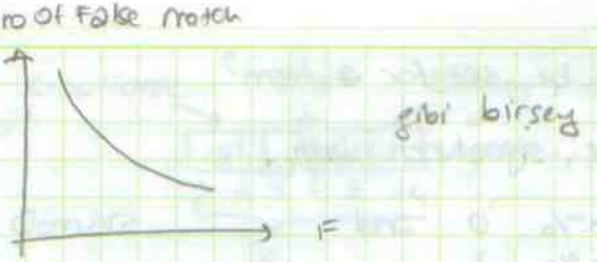
}

S = block signature

↑ Tries to equalize no of 0s and no of 1s in a block signature (equal weight)

D no of distinct terms





How To Calculate No of False Drop (False Drop Probability)

F : Signature size (in bits)
S : % of bits set / term
D : % of terms / block

Qw : Query weight (No of 1s in query)

Assumption : No of True matches << no of Items

False Drop Probability (f_d) : Probability of matching all 1s of the query by chance.

$$f_d = (1 - (1 - S/F)^D)^{Qw}$$

Qw # of 1s we have in the query bit signature

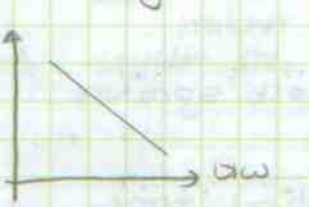
S/F : Probability of setting a bit position by a block term

$(1 - S/F)$: Probability of not setting a bit position by a term

$(1 - S/F)^D$: Probability of not setting a bit position by any of the terms

$(1 - (1 - S/F)^D)$: Probability of setting a bit position by any of the terms

$f_d = (1 - (1 - S/F)^D)^{Qw}$: Probability of setting all query bit positions by chance



INFORMATION FILTERING (IF)

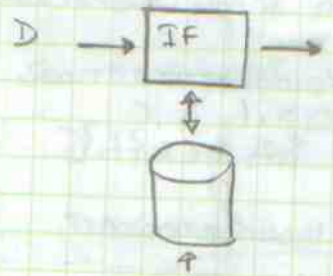
(Selective Dissemination of Information)

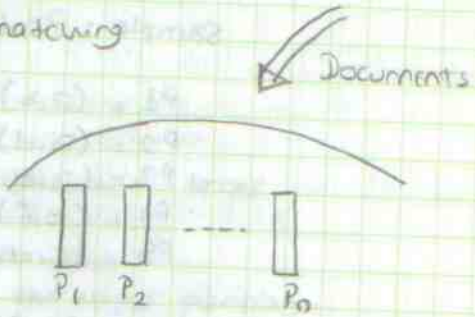
IR / IF

"two sides of the same coin"

Com of the ACM

Belkin & Croft
C ACM 1992





∴ Categorization : supervised clustering

* TDT : Topic Detection & Tracking (TREC Conf.)

TDT Conference

Yan T.W., Garcia Molina H.

"Index Structure for selective dissemination of information

ACM TODS, 1994

Index Structure For IF :

Assume that user profiles are in conjuctive form

t_1 and t_2 and t_3 ...

1- Brute Force Method :

Compare the incoming document with each profile one by one

Use an occurence table

Start with the least frequent word in the profile, the next loss frequent term so on

This makes it possible to detect non matching profiles fast.

Yan, Garcia-Molina Paper → no occurence approach table: Random_Brut Force Method

→ with occurence table Ranked_Brut Force Method

base line — straw man

2. The Counting Method: 01/05/2008

Sample Profiles:

P1 = (a,b)
P2 = (a,d)
P3 = (a,d,e)
P4 = (b,f)
P5 = (c,d,e,f)

Sample Document:
a, c, a, f, b, c
{a, b, c, f}
↑ unique ones

	TOTAL COUNT	a	b	c
P1	2	~~0~~ x 2		
P2	2	~~0~~ 1		
P3	3	~~0~~ 1		
P4	2	0	~~0~~ x 2	
P5	4	0	~~0~~ x 2	

DIRECTORY	DISK Posting List
a	P1 | P2 | P3
b	P1 | P4
c	P5
d	P2 | P3 | P5
e	P3 | P5
f	P4 | P5

The incoming document satisfies P1 P4

For each document there is a disk access for retrieving the related (term) posting list.

THE KEY METHODS

In the counting method, a profile $(\omega_1, \omega_2, \ldots, \omega_k)$ appears in 'k' posting lists.

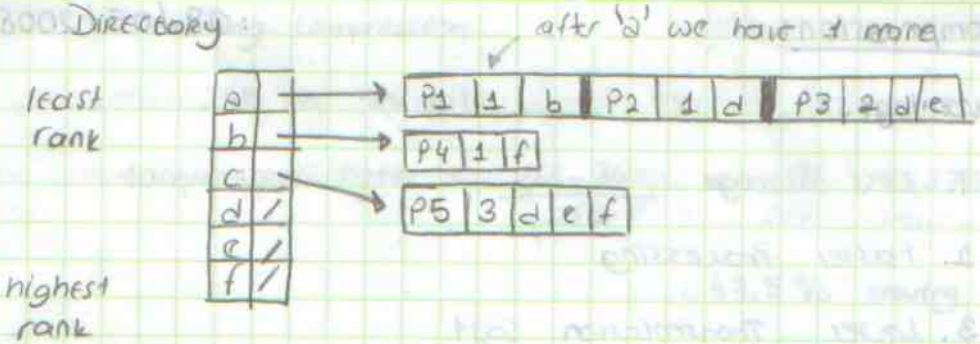
In the key methods a profile only appears in the list of one of its words. That word is called the key.

↙ RANDOM KEY METHOD (any word is the key)

↘ RANKED KEY METHOD (store the profile in the list of the word with the lowest rank)

lowest — least freq in docs

most — highest freq. in docs

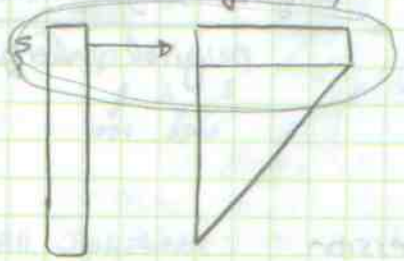


We don't need "COUNT" and "TOTAL" data structures.

Search Algorithm

$D = \{a, b, c, f\}$

Consider all of the posting lists for document terms

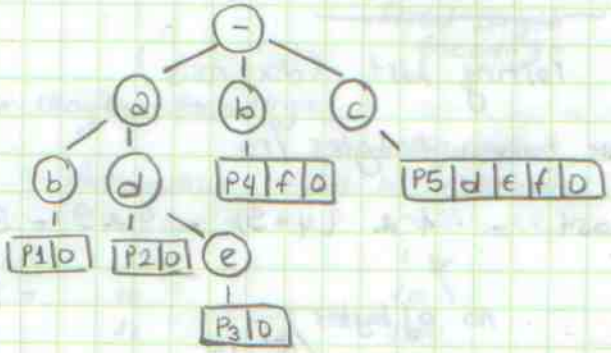


most of the time
we skip the upper portion
of the posting list triangle

↓
least freq. olduğu için
dökümanda geçmeyecek terim
(doc larda çok geçen bir term
olmamasına rağmen, profilleride çok
geçiyor)

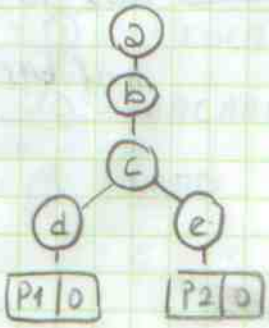
THE TREE METHODS

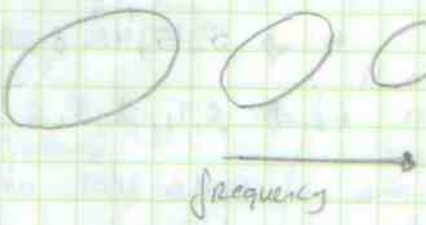
Users with similar interest areas would have similar profiles
(their profiles would have many common terms)



Consider two profiles:

(a, b, c, d)
(a, b, c, e)





Compression: 08/05/2008

Advantages:

1. Less Storage
2. Faster Processing
3. Lower Transmission Cost

Disadvantages:

Compression → uncompress / decompression cost

How to measure compression effectiveness:

$$\text{Compression Ratio} = \frac{\text{Original Length} - \text{Encoded Length}}{\text{Original Length}}$$

Types of compression:

Text compression
Inverted Index Compression
(Posting Lists)

Inverted Index compression:

$t_1 \rightarrow 5, 10, 22, 30$

$t_2 \rightarrow 3, 7, 9, 10, 16$

Posting List (doc no.s)

⇒ Assume that we have 4 bytes /no

Storage cost = 4 * (4+5) = 4 * 9 = 36 bytes

= 36 * 8 = 288 bits

(4 = no of bytes /int no)

Use Run Length Encoding:

$t_1 \Rightarrow 5, 5, \overline{12, 8}$ → max no 12

# of bits needed = $\lceil \log_2 12 \rceil = 14$

$1100_2 = 12_{10}$.

$t_2 \Rightarrow 3, 4, 2, 1, 6$

Cost using Compression:

288 vs. 86 bits

Compression Ratio = $\frac{288 - 36}{288} = \frac{252}{288} = 0{,}875$

87,5 % saving

Text Compression Methods:

SPECIAL PURPOSE COMPRESSION:

1. All numeric fields

123 ⇒ F1 F2 F3 ⇐ EBCDIC representation
1111 0001
Extended Binary Coded Decimal Interchange Code

DROP F's

1 2 3
0001 0010

50 % savings

ASCII
30 31 33
0011 0000

2. All alphabetic

26 letters + blank $\lceil \log_2 27 \rceil = 5$ $2^5 = 32$

27 different characters 5 vs. 8 ⇒ $\frac{5}{8}$ 0,625 ⇒ 37,5% savings

↗ Fixed Length Encoding

Variable Length Encoding:

Char	Ambiguous	unambiguous not instantaneous	Instantaneous
A	1	1	0
B	10	10	10
C	11	100	110
D	100	1000	111

111 00 11 1
C D C A

① COCA or AADAC

② AACAAA → unorb.

③ DAAD no look-a-head

if a char appear more # of times should be represented with less # of bits & vice versa

⇓

for this

⇓

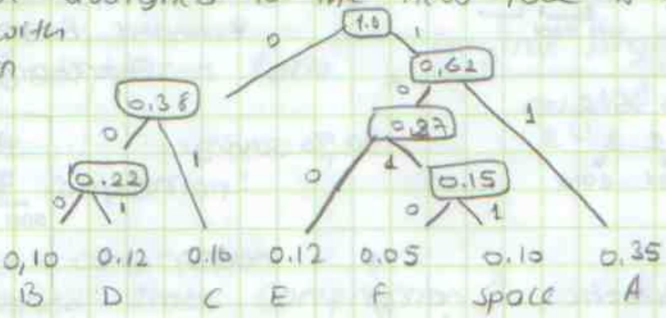
Hauffmann coding

Haufman Coding :

We have to know the relative frequency of the characters.

alphabet
F : 0,05 → only 5% of the characters are F
space : 0,10
B : 0.10
D : 0.12
E : 0.12
C : 0.16
A : 0.35

We construct a tree (Hauffman Coding Tree) in an iterative manner. At each iteration we create a new node having as children two nodes with the smallest possibility values.
The value assigned to the new node is the sum of the values associated with its children



B : 000 (3)
D : 001 (3)
C : 01 (2)
E : 100 (3)
F : 1010 (4)
space : 011 (3)
A : 11 (2)

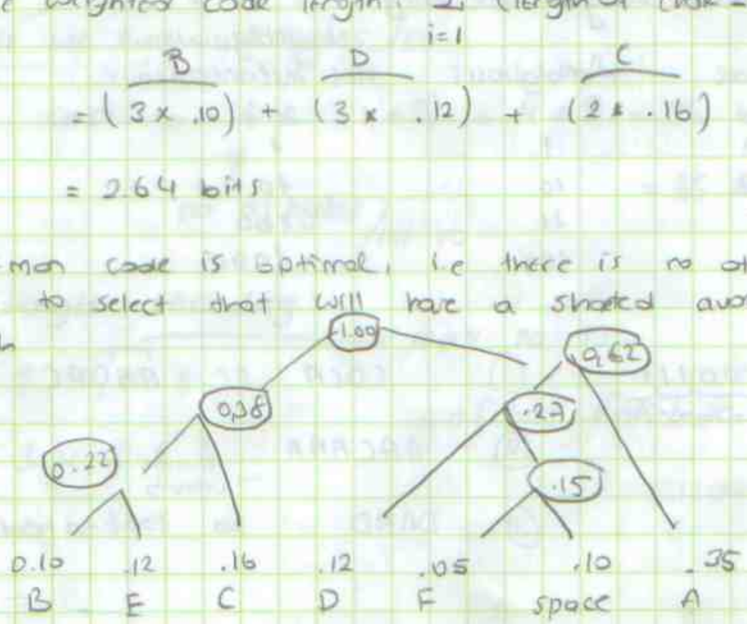
A BAD FEED
11 000 11

The average weighted code length : $\sum_{i=1}^{n}$ (length of char - i) * freq.

$= \underbrace{(3 \times .10)}_{B} + \underbrace{(3 \times .12)}_{D} + \underbrace{(2 \times .16)}_{C} \ldots$

= 2.64 bits

Hauffman code is optimal, i.e there is no other assignment of codes to select that will have a shorted average weighted code length



Please Read!
γ δ delta coding
Bunlar Yazılacak!

NOT : Gamma Beta'dan sorumluyuz!
Ders Notları 'Part-2' den ulaşılabilinir.