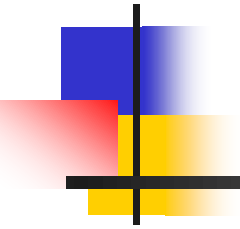


Τεχνητή Νοημοσύνη

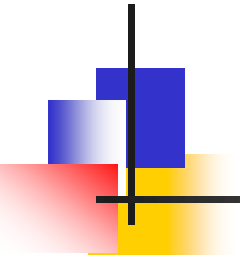
V. Megalooikonomou

Text Analytics

(some slides are based on notes by C. Faloutsos)

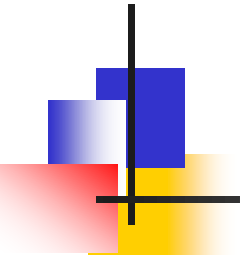
Text analytics- Detailed outline

- text
  - problem
  - full text scanning
  - inversion
  - signature files
  - clustering
  - information filtering and LSI



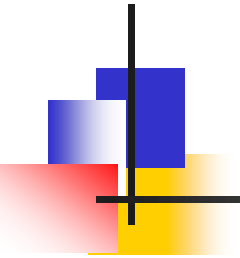
Problem - Motivation

- Eg., find documents containing "*data*", "*retrieval*"
- Applications:



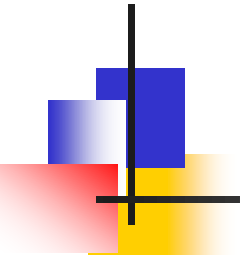
Problem - Motivation

- Eg., find documents containing "*data*", "*retrieval*"
- Applications:
  - Web
  - law + patent offices
  - digital libraries
  - information filtering



Problem - Motivation

- Types of queries:
  - boolean (‘data’ AND ‘retrieval’ AND NOT …)

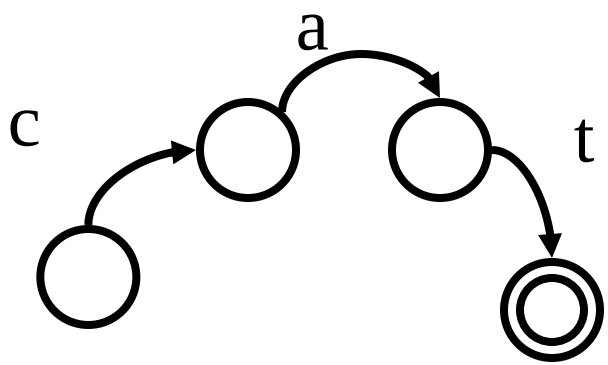


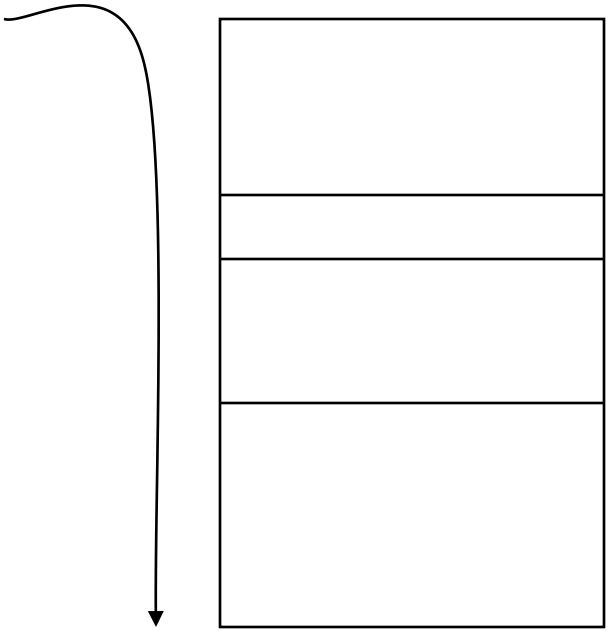
Problem - Motivation

- Types of queries:
  - boolean (‘data’ AND ‘retrieval’ AND NOT ...)
  - additional features (‘data’ ADJACENT ‘retrieval’)
  - keyword queries (‘data’, ‘retrieval’)
- How to search a large collection of documents?

Full-text scanning

- Build a FSA; scan





Full-text scanning

- for single term:
  - (naive: O(N*M))

ABRACADABRA text

CAB pattern

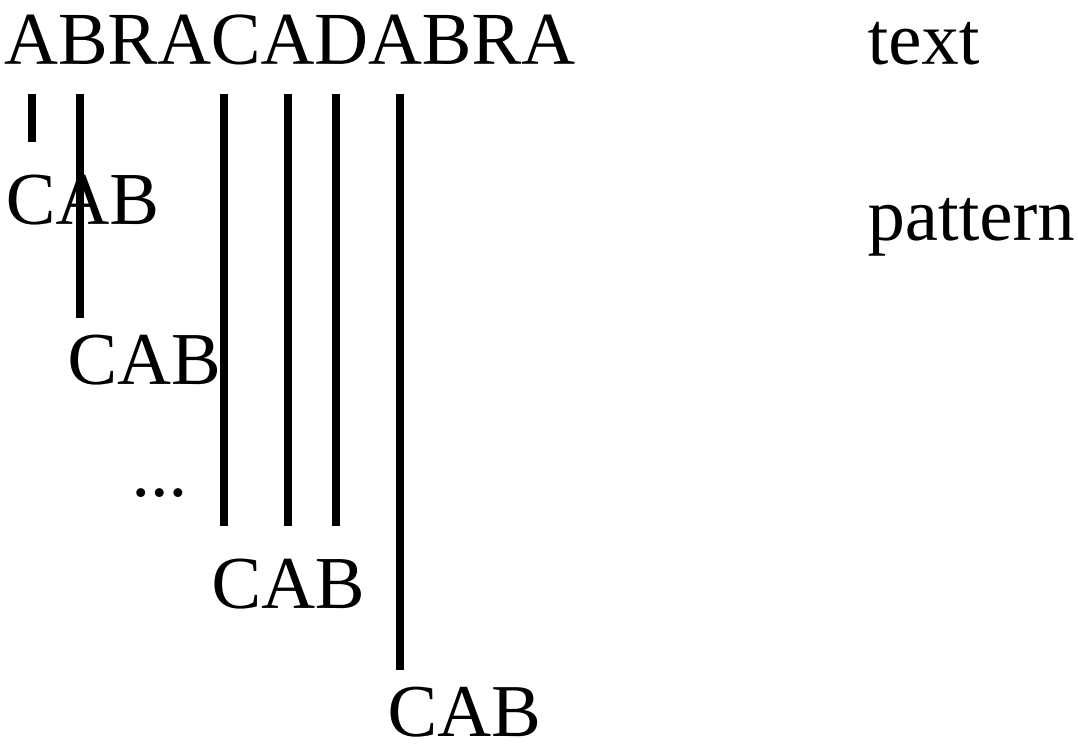
Full-text scanning

- for single term:
  - (naive: O(N*M))
  - Knuth Morris and Pratt (‘77)
    - build a small FSA; visit every text letter once only, by carefully shifting more than one step

ABRACADABRA text

CAB pattern

Full-text scanning



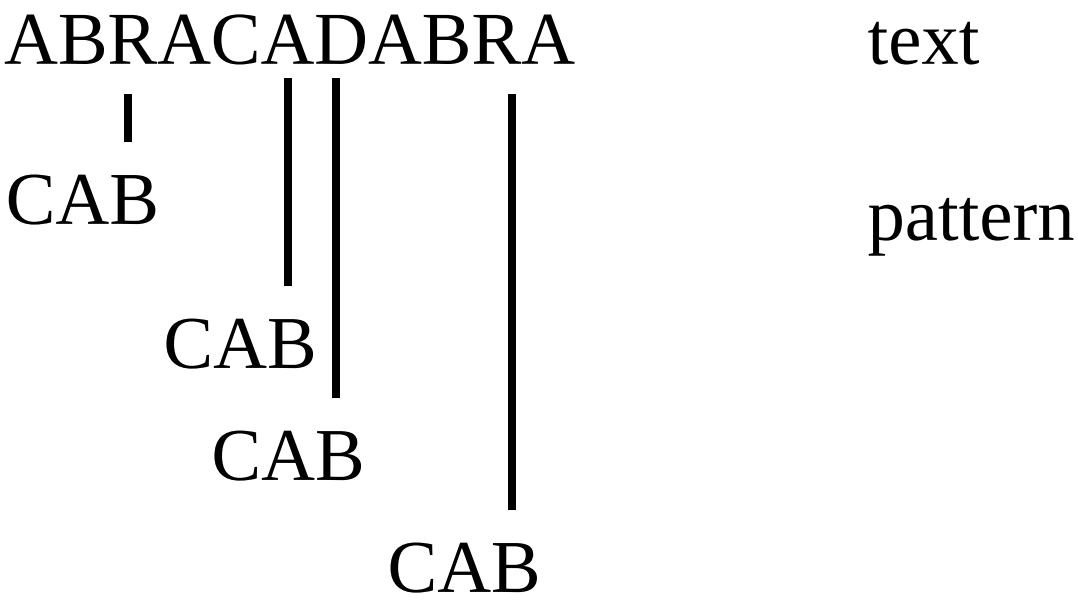
Full-text scanning

- for single term:
  - (naive: O(N*M))
  - Knuth Morris and Pratt (‘77)
  - Boyer and Moore (‘77)
    - preprocess pattern; start from **right to left** & **skip**!

ABRACADABRA text

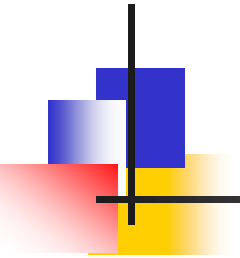
CAB pattern

Full-text scanning



Full-text scanning

Boyer+Moore: fastest, in practice
Sunday (‘90): some improvements



Full-text scanning

- For multiple terms (w/o "don't care" characters): Aho+Corasic ('75)
  - again, build a simplified FSA in O(M) time
- Probabilistic algorithms: 'fingerprints' (Karp + Rabin '87)
- approximate match: 'agrep' [Wu+Manber, Baeza-Yates+, '92]

Full-text scanning

- Approximate matching - **string editing** distance:

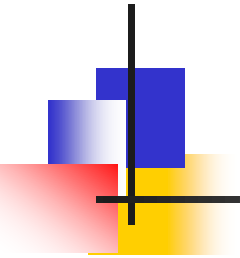
  d(‘survey’, ‘surgery’) = 2

  = min # of insertions, deletions, substitutions to transform the first string

  into the second

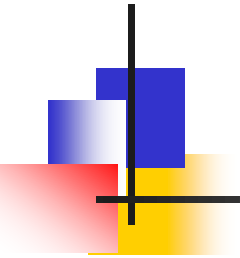
  SURVEY

  SURGERY



Full-text scanning

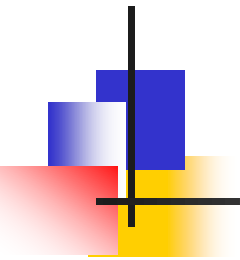
- **string editing** distance - how to compute?
- A:



Full-text scanning

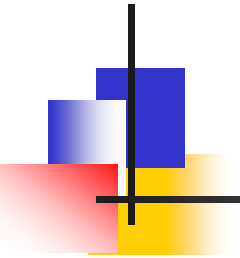
- **string editing** distance - how to compute?
- A: dynamic programming

  cost(i, j) = cost to match prefix of length *i* of first string *s* with prefix of length *j* of second string *t*



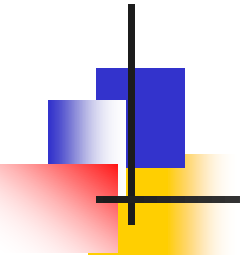
Full-text scanning

```
if s[i] = t[j] then
        cost( i, j ) = cost(i-1, j-1)
else
        cost(i, j ) = min (
                        1 + cost(i, j-1) // deletion
                        1 + cost(i-1, j-1) // substitution
                        1 + cost(i-1, j) // insertion
                            )
```



Full-text scanning

Complexity: O(M*N) (when using a matrix to ‘memorize’ partial results)



Full-text scanning

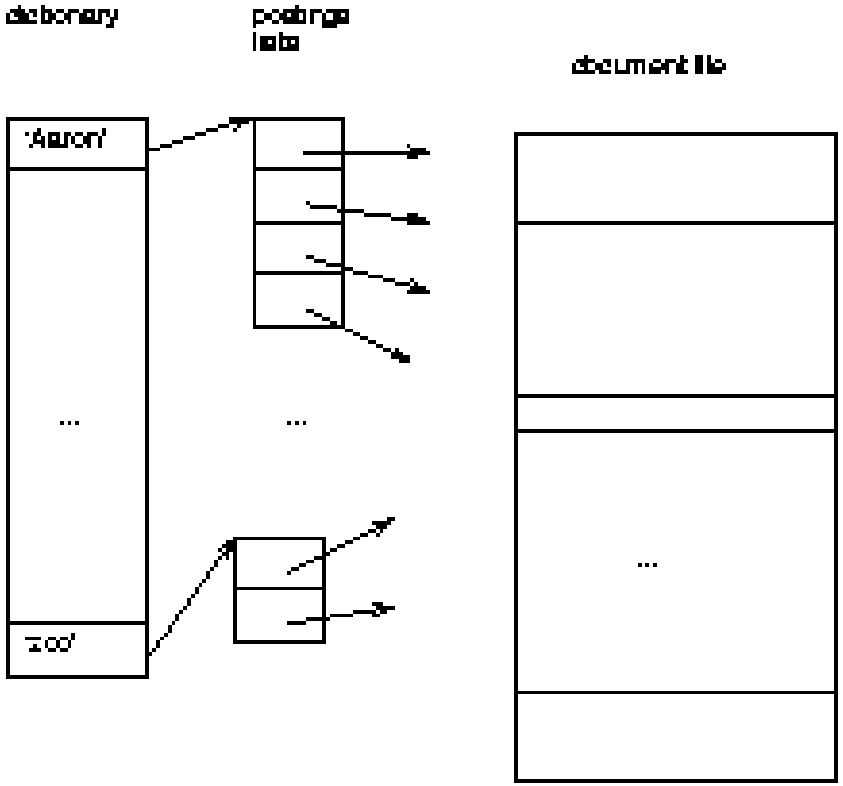
Conclusions:

- Full text scanning needs no space overhead, but is slow for large datasets

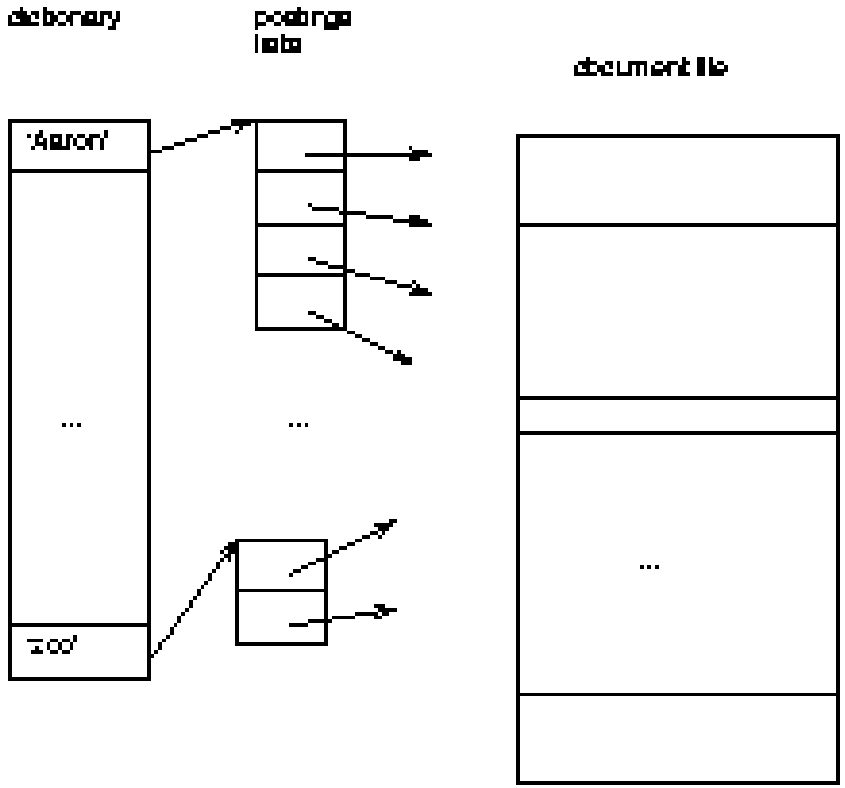
Text analytics- Detailed outline

- text
  - problem
  - full text scanning
  - inversion
  - signature files
  - clustering
  - information filtering and LSI

Text - Inversion

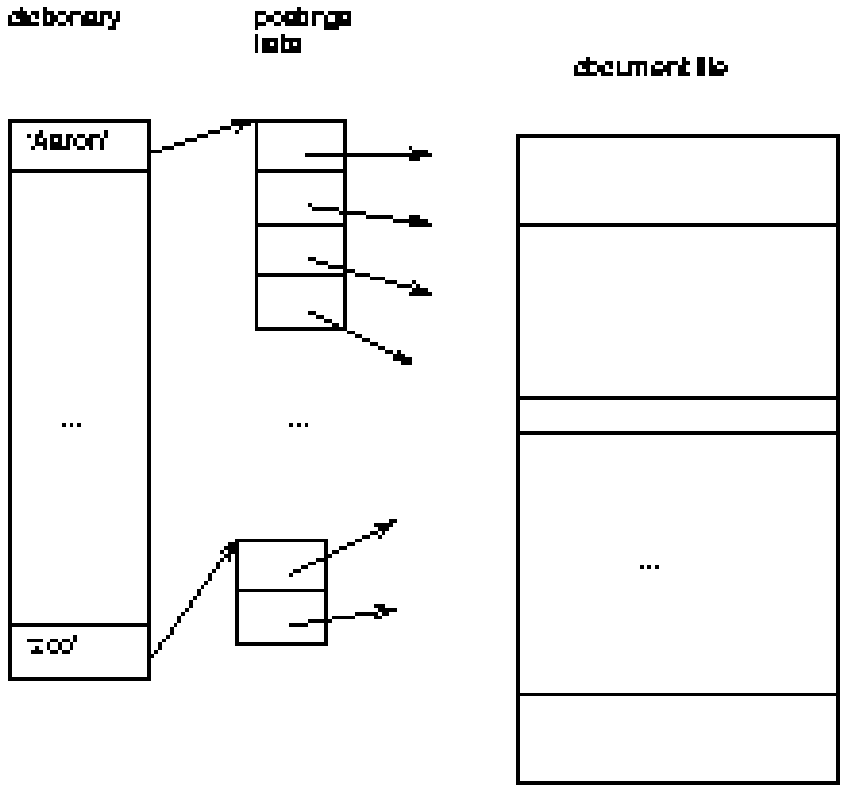


Text - Inversion



Q: space overhead?

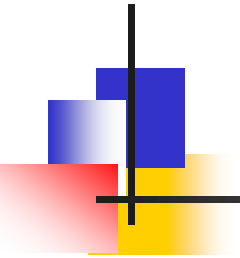
Text - Inversion



A: mainly, the postings lists

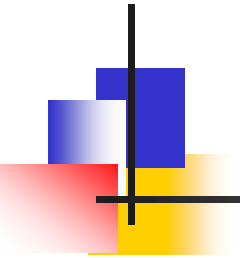
Text - Inversion

- how to organize the dictionary?
- stemming – Y/N?
- insertions?



Text - Inversion

- how to organize the dictionary?
  - B-tree, hashing, TRIEs, PATRICIA trees, ...
- stemming – Y/N?
- insertions?



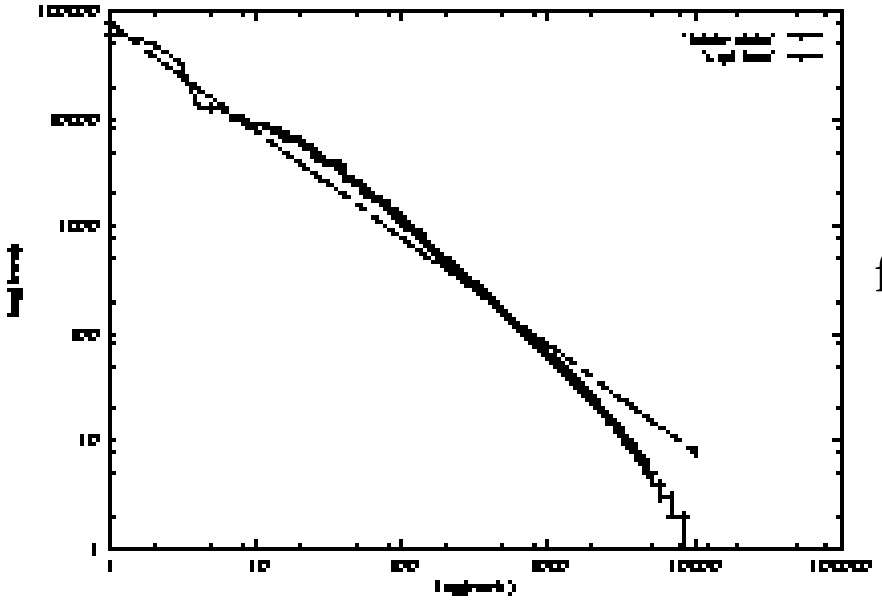
Text – Inversion

- other topics:
  - Parallelism [Tomasic+,93]
  - Insertions [Tomasic+94], [Brown+]
    - ‘zipf’ distributions
  - Approximate searching (‘glimpse’ [Wu+])

Text - Inversion

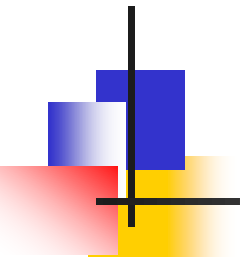
- postings list – more Zipf distr.: eg., rank-frequency plot of 'Bible'

log(freq)



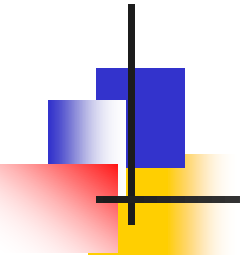
freq ~ 1 / (rank * ln(1.78V))

log(rank)



Text - Inversion

- postings lists
  - Cutting+Pedersen
    - (keep first 4 in B-tree leaves)
  - how to allocate space: [Faloutsos+92]
    - geometric progression
  - compression (Elias codes) [Zobel+] – down to 2% overhead!



Conclusions

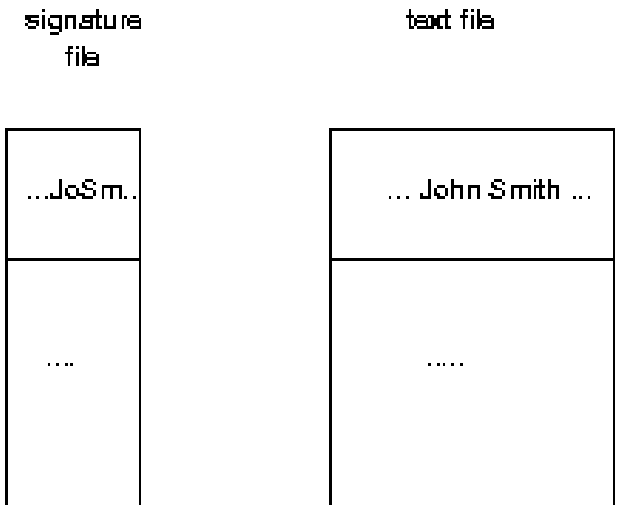
- needs space overhead (2%-300%), but it is the fastest

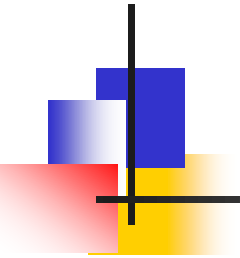
Text analytics- Detailed outline

- text
  - problem
  - full text scanning
  - inversion
  - signature files
  - clustering
  - information filtering and LSI

Signature files

- idea: ‘quick & dirty’ filter





Signature files

- idea: ‘quick & dirty’ filter
- then, do sequential scan on signature file and discard ‘false alarms’
- Adv.: easy insertions; faster than seq. scan
- Disadv.: O(N) search (with small constant)

- Q: how to extract signatures?

Signature files

- A: superimposed coding!! [Mooers49], ...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

m (=4 bits/word) ~ (=4 bits set to “1” and the rest left as “0”)
F (=12 bits sign. size)
the bit patterns are OR-ed to form the document signature

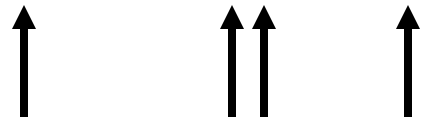
Signature files

- A: superimposed coding!! [Mooers49],

...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

data



actual match

Signature files

- A: superimposed coding!! [Mooers49],

...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

retrieval

actual dismissal

Signature files

- A: superimposed coding!! [Mooers49],

...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011
nucleotic	↑ ↑ ↑ ↑

false alarm (‘false drop’)

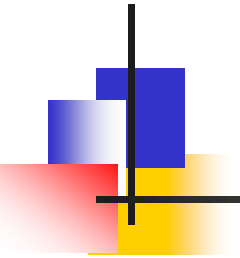
Signature files

- A: superimposed coding!! [Mooers49],

...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

'YES' is 'MAYBE'
'NO' is 'NO'



Signature files

- Q1: How to choose F and m ?
- Q2: Why is it called ‘false drop’?
- Q3: other apps of signature files?

Signature files

- Q1: How to choose *F* and *m* ?

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

m (=4 bits/word)
F (=12 bits sign. size)

Signature files

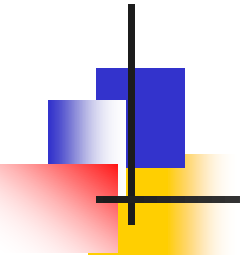
- Q1: How to choose *F* and *m* ?
- A: so that doc. signature is 50% full

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

m (=4 bits/word)
F (=12 bits sign. size)

Signature files

- Q1: How to choose F and m ?
- Q2: Why is it called 'false drop'?
- Q3: other apps of signature files?

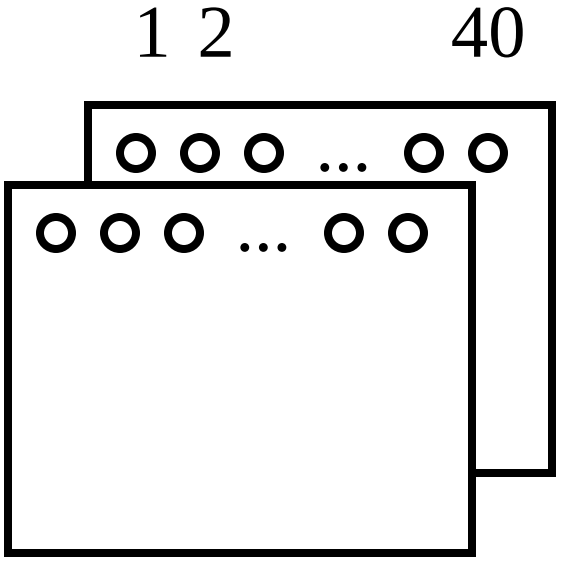


Signature files

- Q2: Why is it called ‘false drop’?
- Very old but fascinating story [1949]
  - how to find qualifying books (by title word, and/or author, and/or keyword)
  - in O(1) time?
  - *without computers*

Signature files

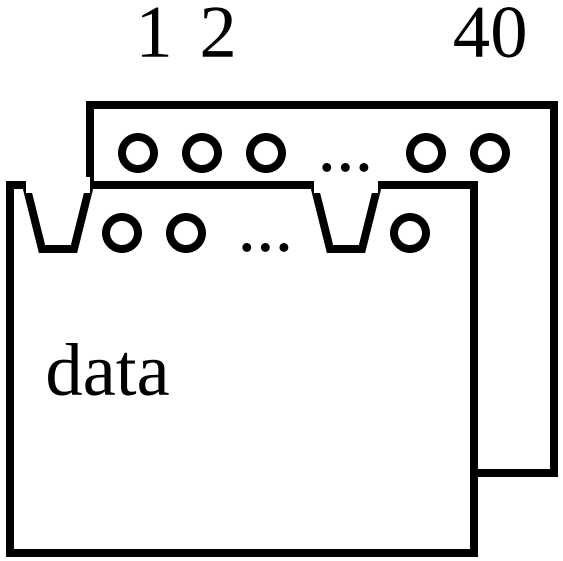
- Solution: edge-notched cards



•each title word is mapped to *m* numbers(how?)
•and the corresponding holes are cut out:

Signature files

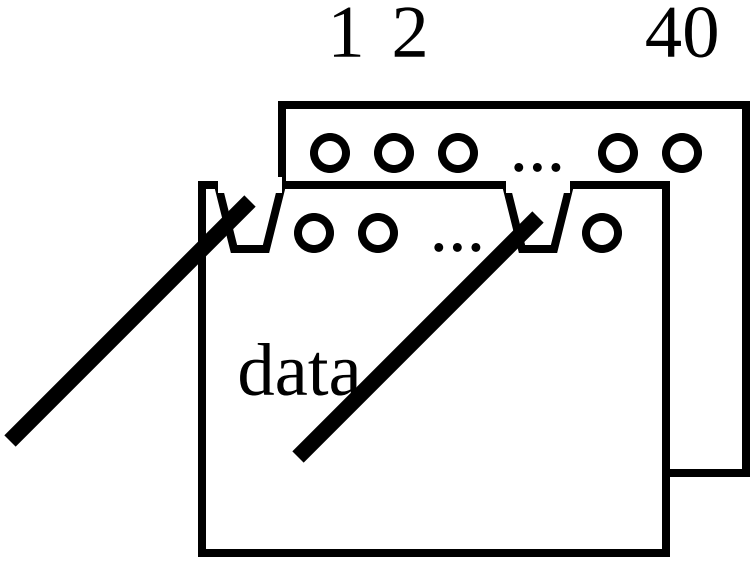
- Solution: edge-notched cards



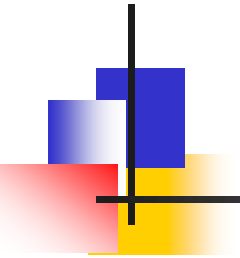
‘data’ -> #1, #39

Signature files

- Search, e.g., for ‘data’: activate needle #1, #39, and shake the stack of cards!

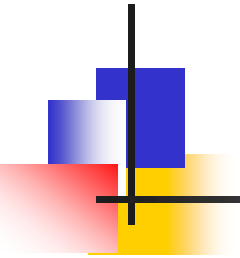


‘data’ -> #1, #39



Signature files

- Also known as ‘zatocoding’, from ‘Zator’ company.



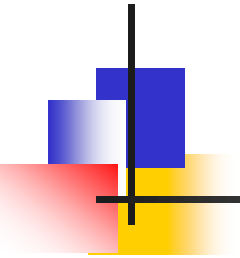
Signature files

- Q1: How to choose F and m ?
- Q2: Why is it called ‘false drop’?
- Q3: other apps of signature files?

Signature files

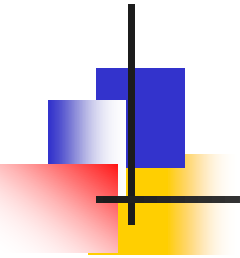
- Q3: other apps of signature files?
- A: anything that has to do with ‘membership testing’: does ‘data’ belong to the set of words of the document?

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011



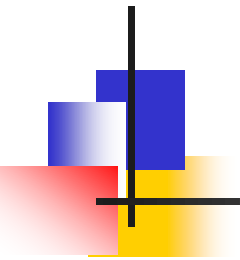
Signature files

- UNIX's early 'spell' system [McIlroy]
- Bloom-joins in System R* [Mackert+] and 'active disks' [Riedel99]
- differential files [Severance+Lohman]



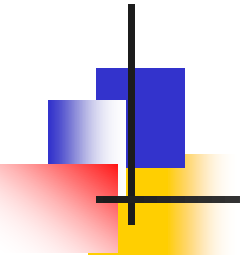
Signature files - conclusions

- easy insertions; slower than inversion
- brilliant idea of ‘quick and dirty’ filter: quickly discard the vast majority of non-qualifying elements, and focus on the rest.



References

- Aho, A. V. and M. J. Corasick (June 1975). "Fast Pattern Matching: An Aid to Bibliographic Search." CACM 18(6): 333-340.
- Boyer, R. S. and J. S. Moore (Oct. 1977). "A Fast String Searching Algorithm." CACM 20(10): 762-772.
- Brown, E. W., J. P. Callan, et al. (March 1994). Supporting Full-Text Information Retrieval with a Persistent Object Store. Proc. of EDBT conference, Cambridge, U.K., Springer Verlag.
- Faloutsos, C. and H. V. Jagadish (Aug. 23-27, 1992). On B-tree Indices for Skewed Distributions. 18th VLDB Conference, Vancouver, British Columbia.
- Karp, R. M. and M. O. Rabin (March 1987). "Efficient Randomized Pattern-Matching Algorithms." IBM Journal of Research and Development 31(2): 249-260.
- Knuth, D. E., J. H. Morris, et al. (June 1977). "Fast Pattern Matching in Strings." SIAM J. Comput 6(2): 323-350.



References - cont'd

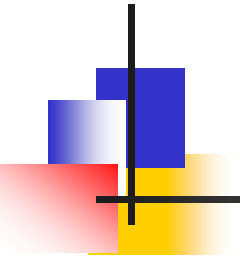
- Mackert, L. M. and G. M. Lohman (August 1986). R* Optimizer Validation and Performance Evaluation for Distributed Queries. Proc. of 12th Int. Conf. on Very Large Data Bases (VLDB), Kyoto, Japan.
- Manber, U. and S. Wu (1994). GLIMPSE: A Tool to Search Through Entire File Systems. Proc. of USENIX Techn. Conf.
- McIlroy, M. D. (Jan. 1982). "Development of a Spelling List." IEEE Trans. on Communications COM-30(1): 91-99.
- Mooers, C. (1949). Application of Random Codes to the Gathering of Statistical Information
- Bulletin 31. Cambridge, Mass, Zator Co.
- Pedersen, D. C. a. J. (1990). Optimizations for dynamic inverted index maintenance. ACM SIGIR.
- Riedel, E. (1999). Active Disks: Remote Execution for Network Attached Storage. ECE, CMU. Pittsburgh, PA.

References - cont'd

- Severance, D. G. and G. M. Lohman (Sept. 1976). "Differential Files: Their Application to the Maintenance of Large Databases." ACM TODS 1(3): 256-267.
- Tomasic, A. and H. Garcia-Molina (1993). Performance of Inverted Indices in Distributed Text Document Retrieval Systems. PDIS.
- Tomasic, A., H. Garcia-Molina, et al. (May 24-27, 1994). Incremental Updates of Inverted Lists for Text Document Retrieval. ACM SIGMOD, Minneapolis, MN.
- Wu, S. and U. Manber (1992). "AGREP- A Fast Approximate Pattern-Matching Tool." .
- Zobel, J., A. Moffat, et al. (Aug. 23-27, 1992). An Efficient Indexing Technique for Full-Text Database Systems. VLDB, Vancouver, B.C., Canada.

Text analytics- Detailed outline

- text
  - problem
  - full text scanning
  - inversion
  - signature files
  - clustering
  - information filtering and LSI

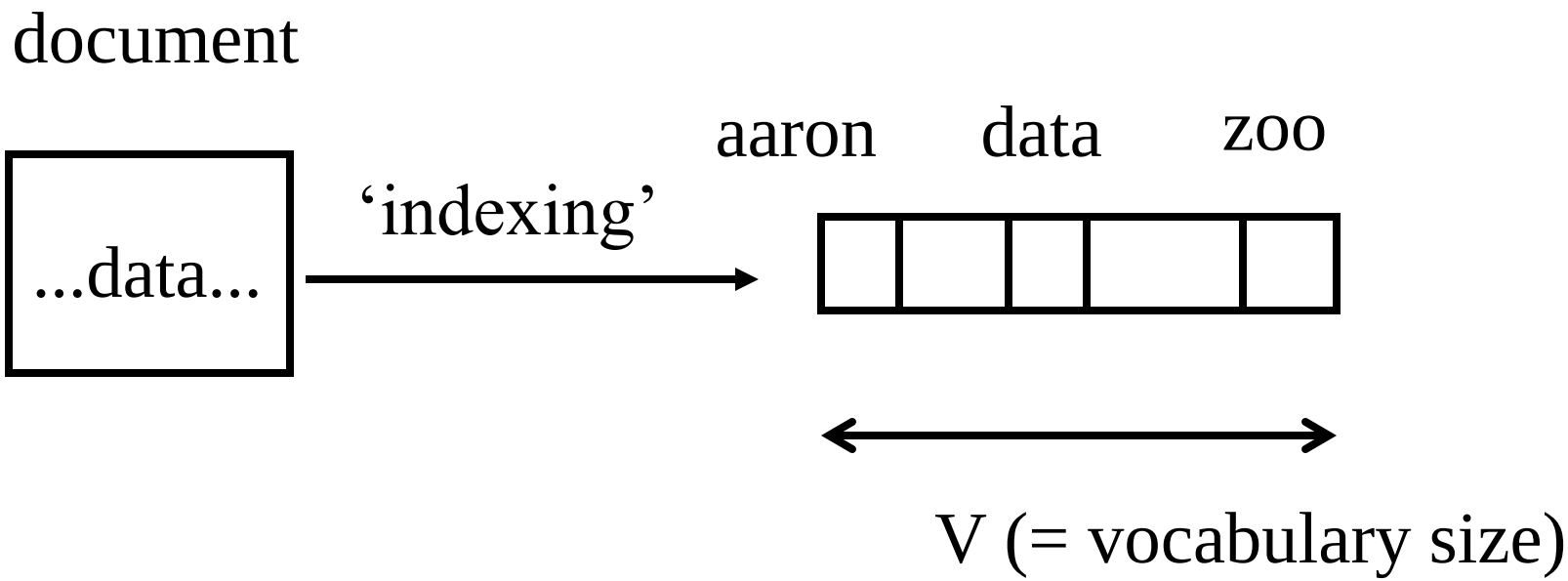


Vector Space Model and Clustering

- keyword queries (vs Boolean)
- each document: -> vector (HOW?)
- each query: -> vector
- search for ‘similar’ vectors

Vector Space Model and Clustering

- main idea:



Vector Space Model and Clustering

Then, group nearby vectors together

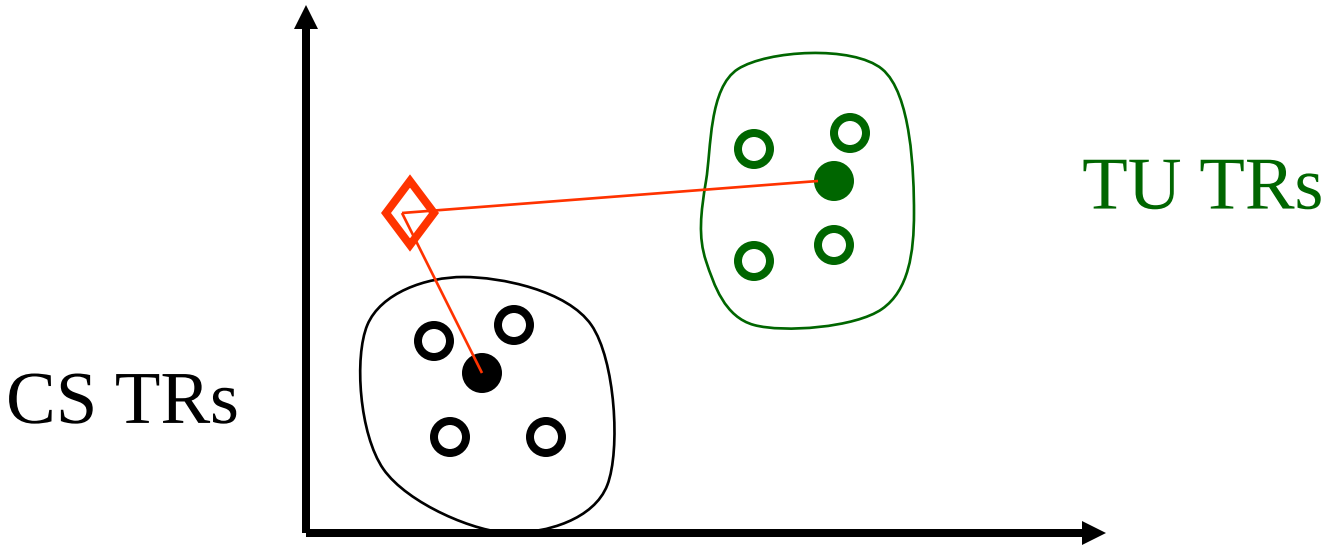
- Q1: cluster search?
- Q2: cluster generation?

Two significant contributions:

- ranked output
- relevance feedback

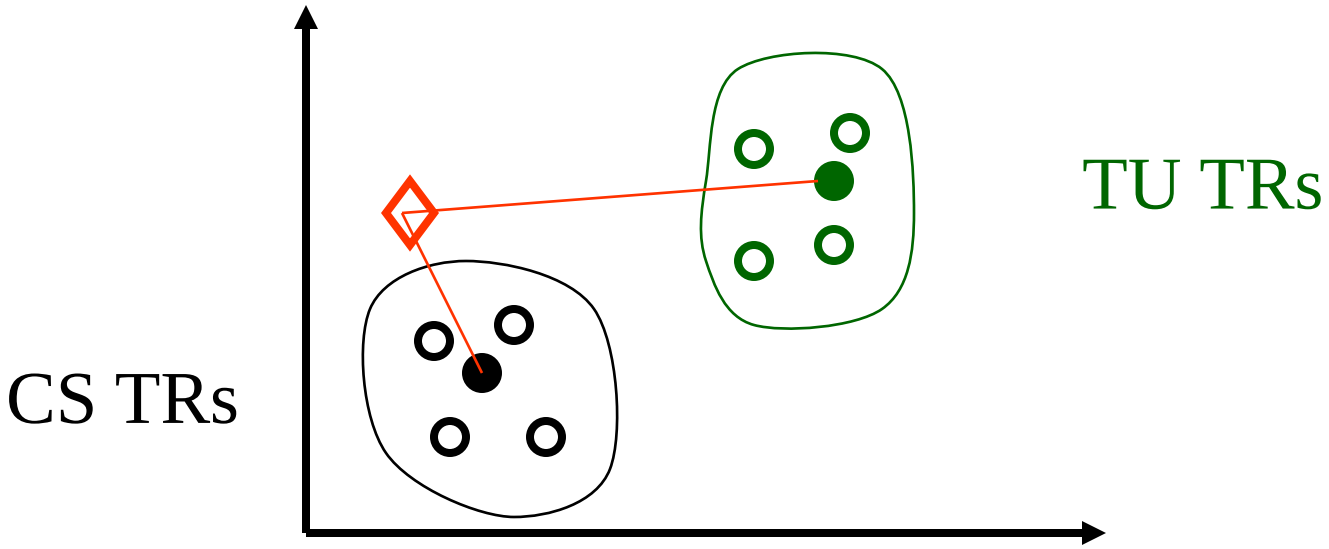
Vector Space Model and Clustering

- cluster search: visit the (k) closest superclusters; continue recursively



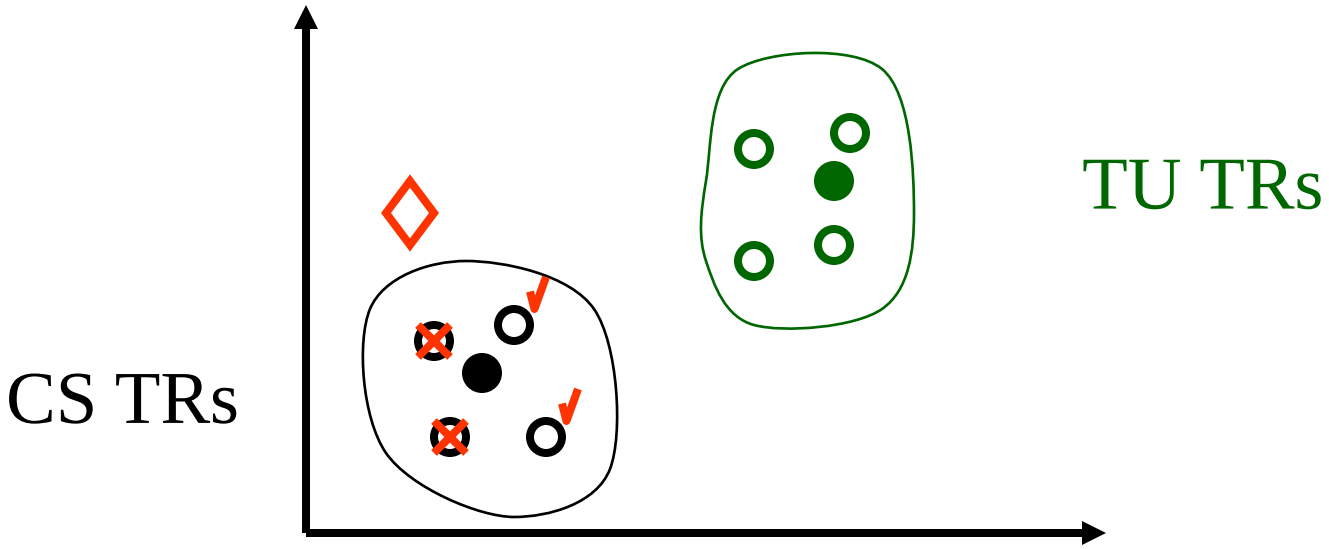
Vector Space Model and Clustering

- ranked output: easy!



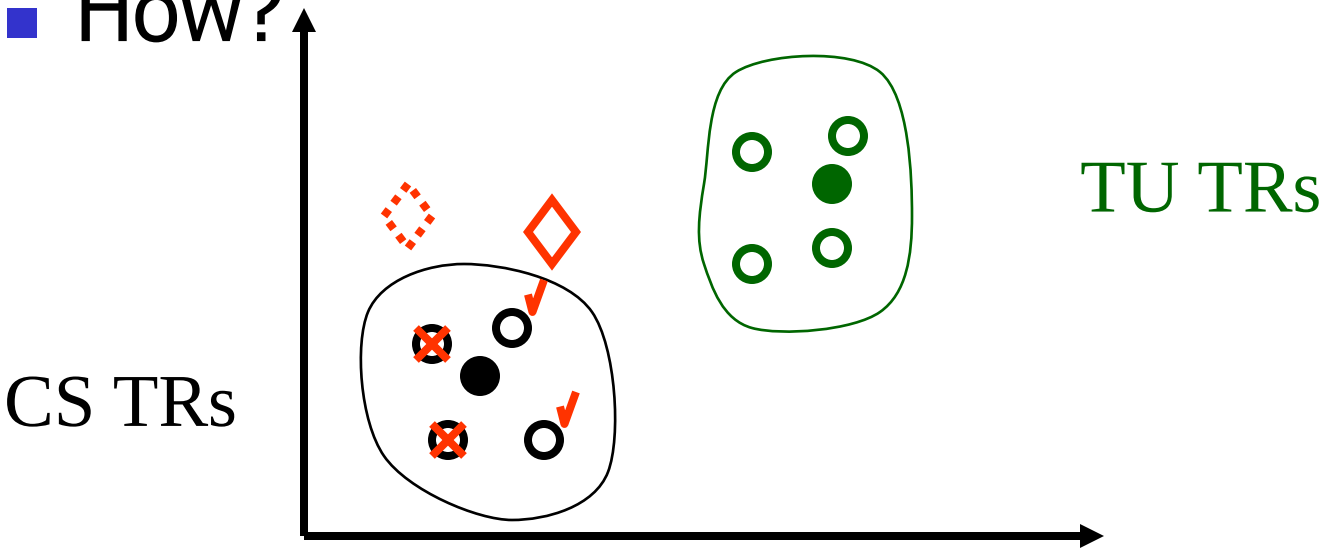
Vector Space Model and Clustering

- relevance feedback (brilliant idea) [Roccio'73]



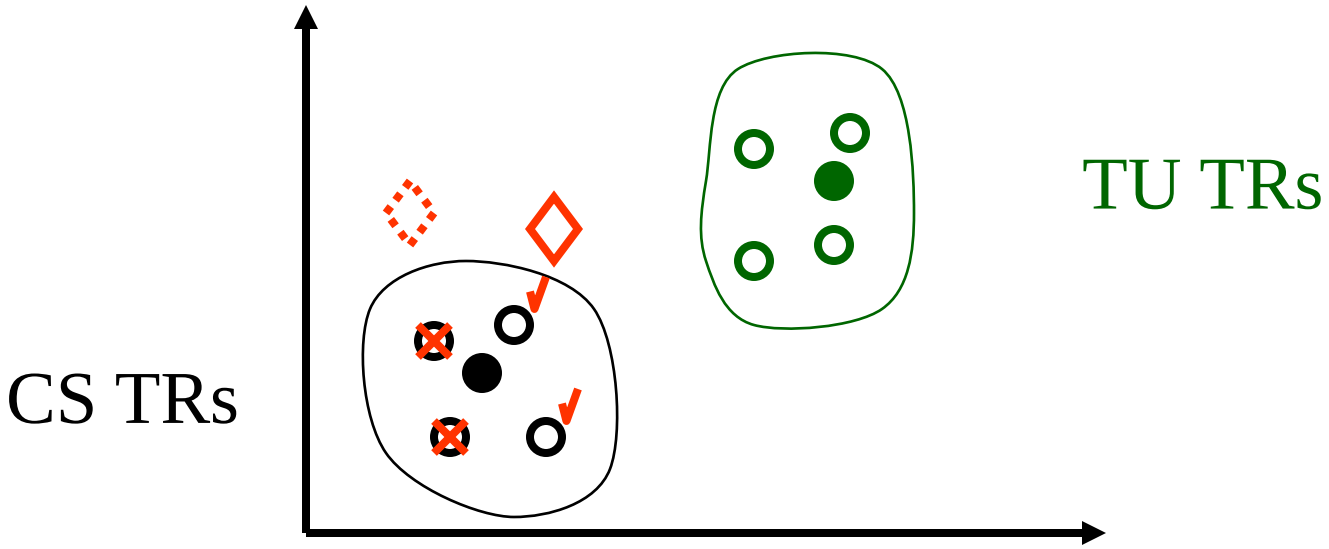
Vector Space Model and Clustering

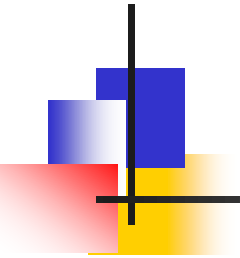
- relevance feedback (brilliant idea) [Roccio'73]
- How?



Vector Space Model and Clustering

- How? A: by adding the 'good' vectors and subtracting the 'bad' ones



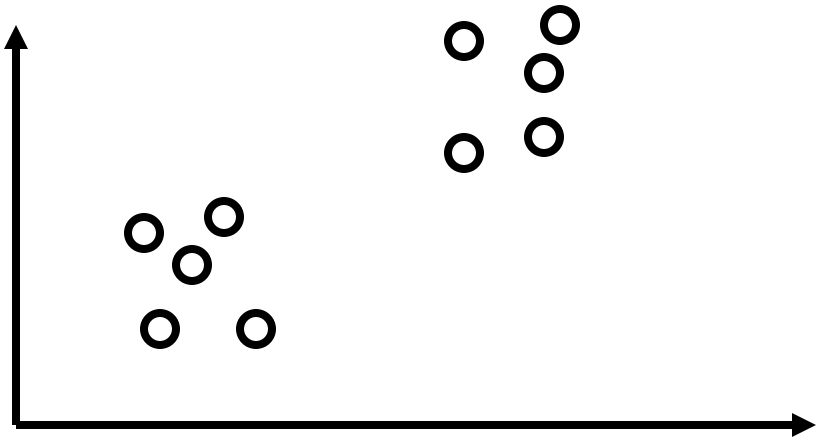


Outline - detailed

- main idea
- cluster search
- cluster generation
- evaluation

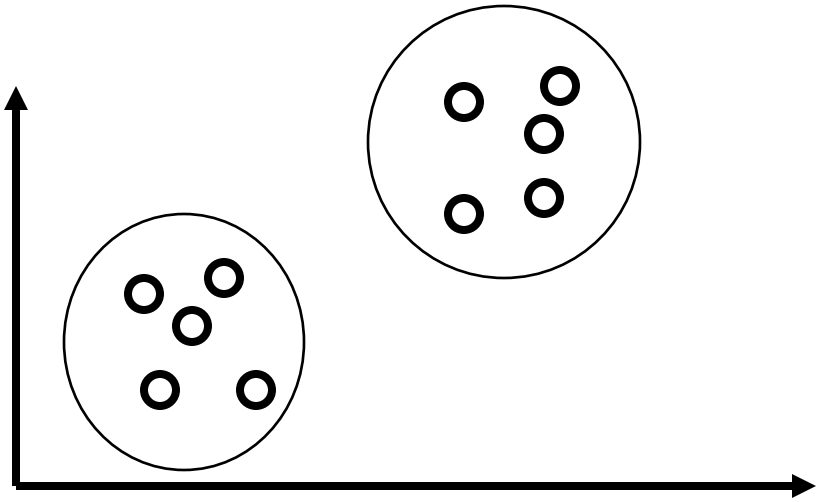
Cluster generation

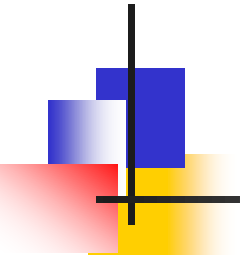
- Problem:
  - given N points in V dimensions, group them



Cluster generation

- Problem:
  - given N points in V dimensions, group them

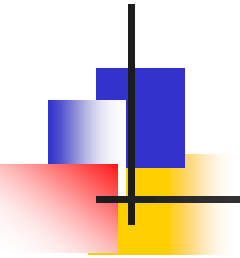




Cluster generation

We need

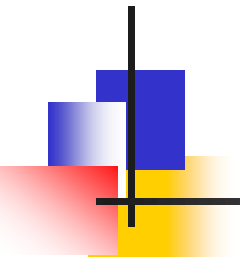
- Q1: document-to-document similarity
- Q2: document-to-cluster similarity



Cluster generation

Q1: document-to-document similarity
(recall: ‘bag of words’ representation)

- D1: {‘data’, ‘retrieval’, ‘system’}
- D2: {‘lung’, ‘pulmonary’, ‘system’}
- distance/similarity functions?



Cluster generation

A1: # of words in common

A2: …….. normalized by the vocabulary sizes

A3: …. etc

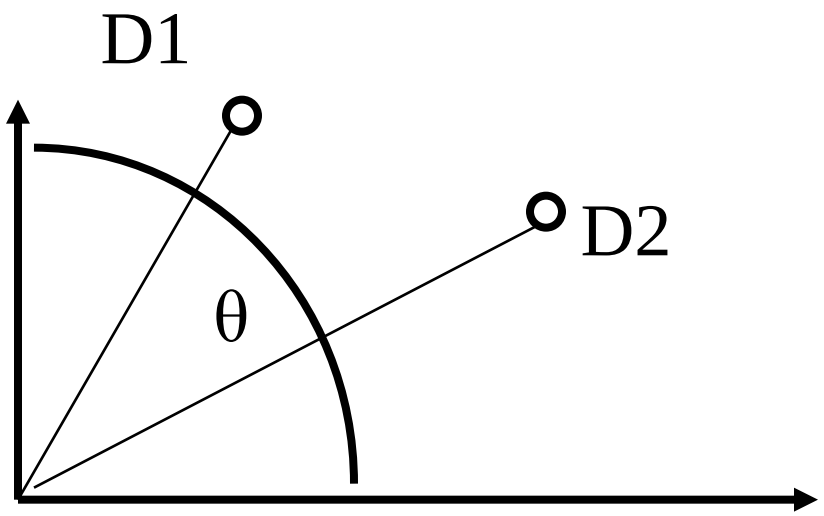
About the same performance - prevailing one:

cosine similarity

Cluster generation

cosine similarity:

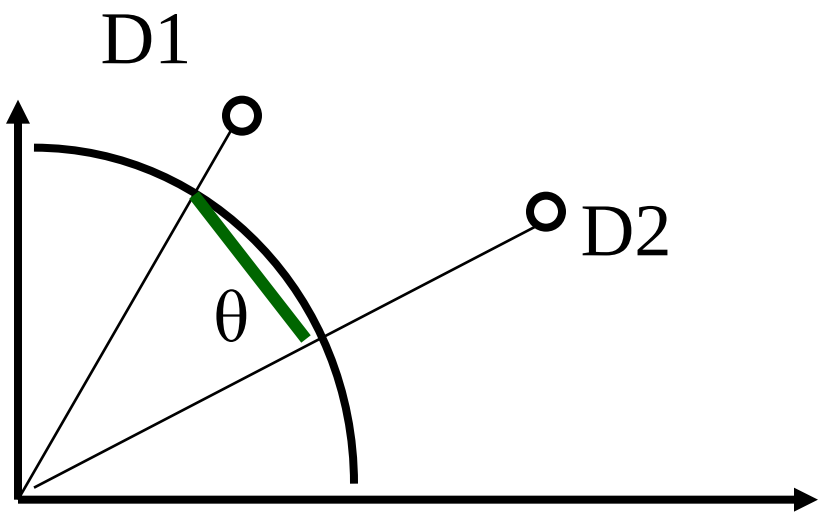
similarity(D1, D2) = cos(θ) = sum($v_{1,i}$ * $v_{2,i}$) / len(v_1)/ len(v_2)

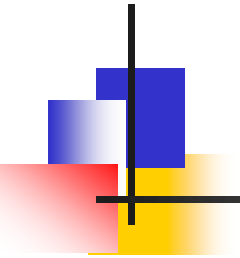


Cluster generation

cosine similarity - observations:

- related to the Euclidean distance
- weights $v_{i,j}$: according to tf/idf





Cluster generation

tf (‘term frequency’)

- high, if the term appears very often in this document.

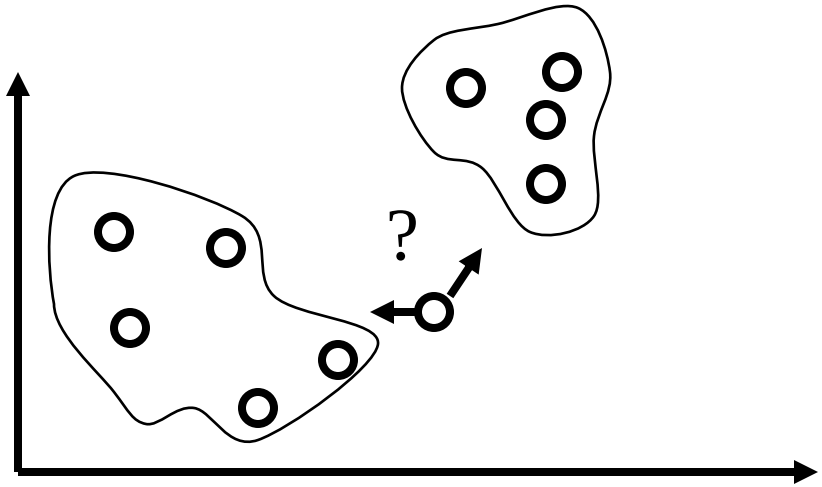
idf (‘inverse document frequency’)

- penalizes ‘common’ words, that appear in almost every document

Cluster generation

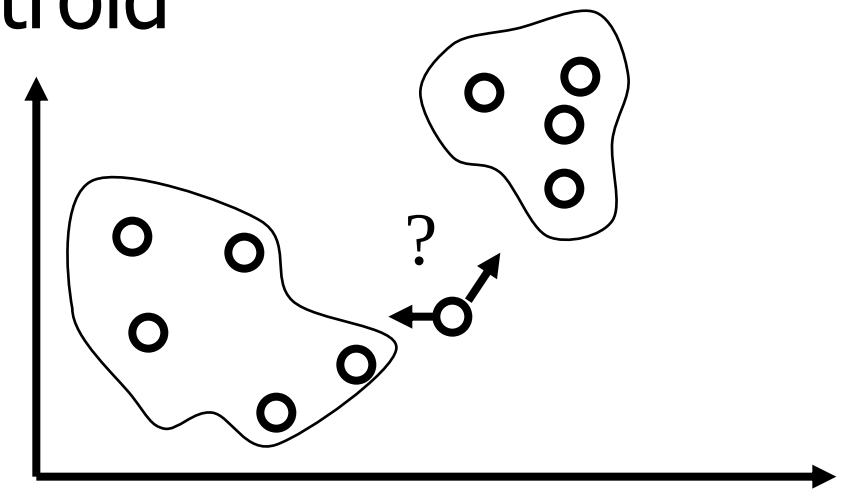
We need

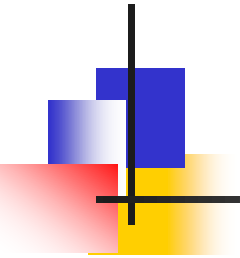
- Q1: document-to-document similarity
- Q2: document-to-cluster similarity



Cluster generation

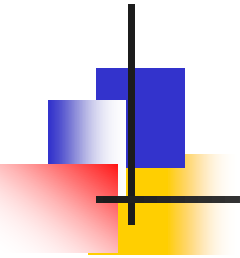
- A1: min distance ('single-link')
- A2: max distance ('all-link')
- A3: avg distance
- A4: distance to centroid





Cluster generation

- A1: min distance ('single-link')
  - leads to elongated clusters
- A2: max distance ('all-link')
  - many, small, tight clusters
- A3: avg distance
  - in between the above
- A4: distance to centroid
  - fast to compute

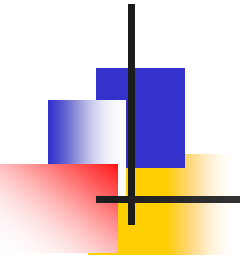


Cluster generation

We have

- document-to-document similarity
- document-to-cluster similarity

Q: How to group documents into ‘natural’ clusters



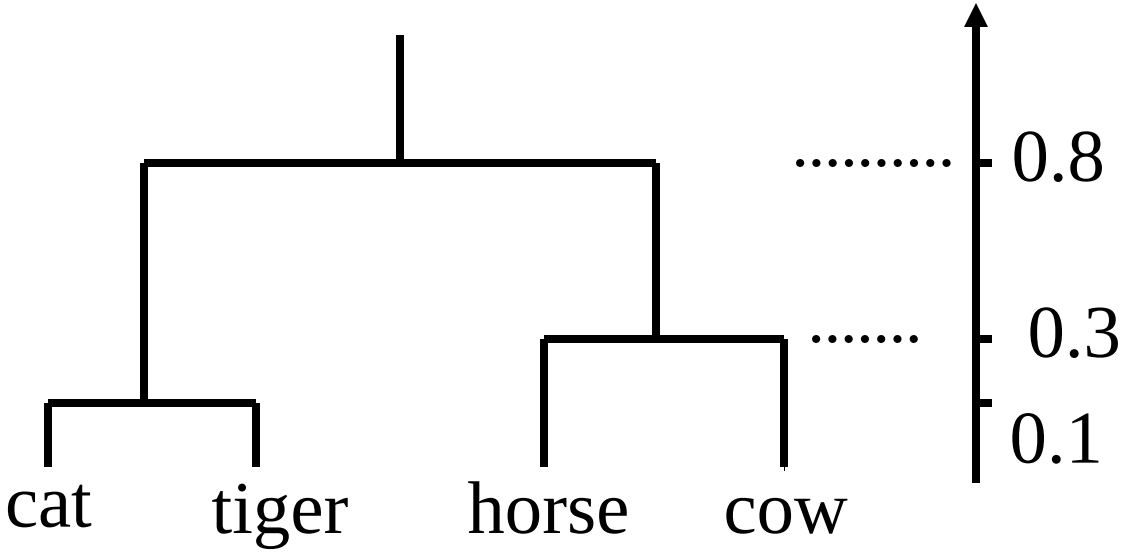
Cluster generation

A: *many-many* algorithms - in two groups [VanRijsbergen]:

- theoretically sound (O(*N^2*))
  - independent of the insertion order
- iterative (O(*N*), O(*N* log(*N*))

Cluster generation - 'sound' methods

- Approach#1:
  - dendrograms - create a hierarchy (bottom up or top-down) - choose a cut-off (how?) and cut

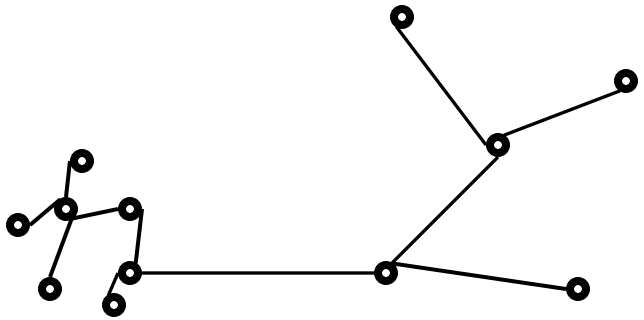


Cluster generation - ‘sound’ methods

- Approach#2:
  - minimize some statistical criterion (eg., sum of squares from cluster centers)
    - like ‘k-means’
    - but how to decide ‘k’?

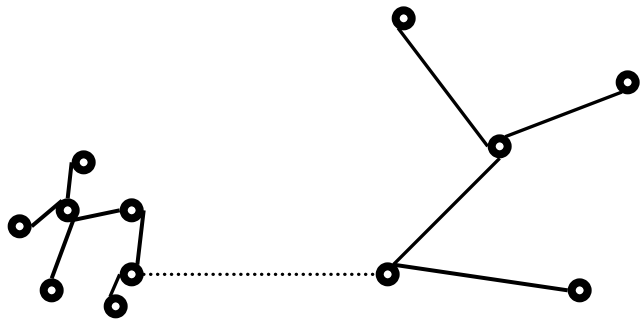
Cluster generation - ‘sound’ methods

- Approach#3:
  - Graph theoretic [Zahn]:
    - build MST;
    - delete edges longer than 2.5* std of the local average



Cluster generation - ‘sound’ methods

- Result:



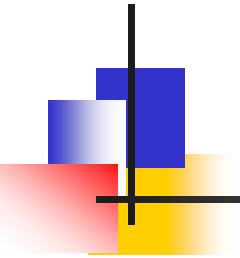
- variations
- Complexity?

Cluster generation - ‘iterative’ methods

General outline:

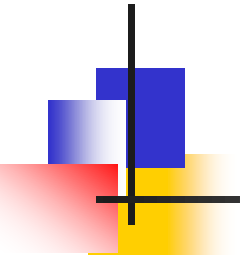
- Choose ‘seeds’ (how?)
- assign each vector to its closest seed (possibly adjusting cluster centroid)
- possibly, re-assign some vectors to improve clusters

Fast and practical, but ‘unpredictable’



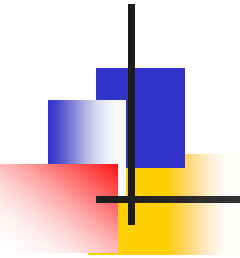
Cluster generation

one way to estimate # of clusters *k* : the 'cover coefficient' [Can+] ~ SVD



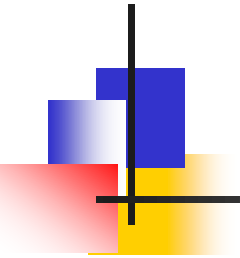
Outline - detailed

- main idea
- cluster search
- cluster generation
- evaluation



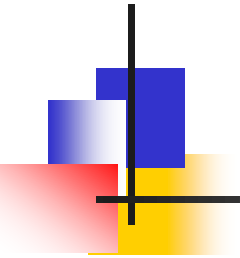
Evaluation

- Q: how to measure ‘goodness’ of one distance function vs another?
- A: ground truth (by humans) and
  - ‘precision’ and ‘recall’



Evaluation

- precision = (retrieved & relevant) / retrieved
  - 100% precision -> no false alarms
- recall = (retrieved & relevant)/ relevant
  - 100% recall -> no false dismissals

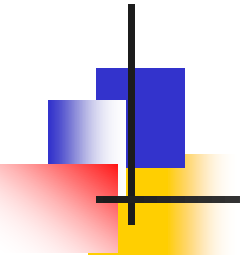


References

- Can, F. and E. A. Ozkarahan (Dec. 1990). "Concepts and Effectiveness of the Cover-Coefficient-Based Clustering Methodology for Text Databases." ACM TODS 15(4): 483-517.
- Noreault, T., M. McGill, et al. (1983). A Performance Evaluation of Similarity Measures, Document Term Weighting Schemes and Representation in a Boolean Environment. Information Retrieval Research, Butterworths.
- Rocchio, J. J. (1971). Relevance Feedback in Information Retrieval. The SMART Retrieval System - Experiments in Automatic Document Processing. G. Salton. Englewood Cliffs, New Jersey, Prentice-Hall Inc.
- Salton, G. (1971). The SMART Retrieval System - Experiments in Automatic Document Processing. Englewood Cliffs, New Jersey, Prentice-Hall Inc.
- Salton, G. and M. J. McGill (1983). Introduction to Modern Information Retrieval, McGraw-Hill.
- Van-Rijsbergen, C. J. (1979). Information Retrieval. London, England, Butterworths.
- Zahn, C. T. (Jan. 1971). "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters." IEEE Trans. on Computers C-20(1): 68-86.

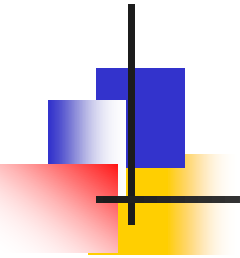
Text analytics- Detailed outline

- text
  - problem
  - full text scanning
  - inversion
  - signature files
  - clustering
  - information filtering and LSI



LSI - Detailed outline

- LSI
  - problem definition
  - main idea
  - experiments



Information Filtering + LSI

- [Foltz+,'92] Goal:
  - users specify interests (= keywords)
  - system alerts them, on suitable news-documents
- Major contribution: LSI = Latent Semantic Indexing
  - latent ('hidden') concepts

Information Filtering + LSI

Main idea:

- map each document into some 'concepts'
- map each term into some 'concepts'

'Concept':~ a set of terms, with weights, e.g.

- "data" (0.8), "system" (0.5), "retrieval" (0.6) -> DBMS_concept

Information Filtering + LSI

Pictorially: term-document matrix (BEFORE)

	'data'	'system'	'retrieval'	'lung'	'ear'
TR1	1	1	1		
TR2	1	1	1		
TR3				1	1
TR4				1	1

Information Filtering + LSI

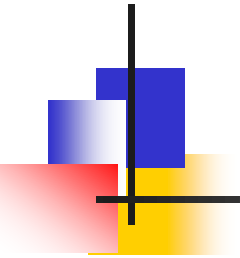
Pictorially: concept-document matrix and...

	'DBMS-concept'	'medical-concept'
TR1	1	
TR2	1	
TR3		1
TR4		1

Information Filtering + LSI

... and concept-term matrix

	'DBMS-concept'	'medical-concept'
data	1	
system	1	
retrieval	1	
lung		1
ear		1



Information Filtering + LSI

Q: How to search, eg., for ‘system’?

Information Filtering + LSI

A: find the corresponding concept(s); and the corresponding documents

	'DBMS-concept'	'medical-concept'
data	1	
system	1	
retrieval	1	
lung		1
ear		1

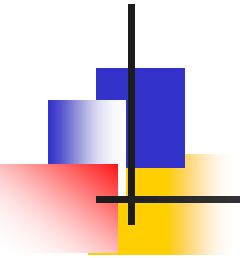
	'DBMS-concept'	'medical-concept'
TR1	1	
TR2	1	
TR3		1
TR4		1

Information Filtering + LSI

A: find the corresponding concept(s); and the corresponding documents

	'DBMS-concept'	'medical-concept'
data	1	
system	1	
retrieval	1	
lung		1
ear		1

	'DBMS-concept'	'medical-concept'
TR1	1	
TR2	1	
TR3		1
TR4		1



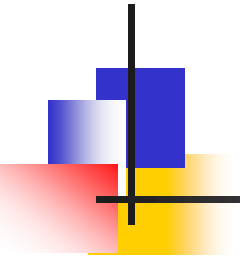
Information Filtering + LSI

Thus it works like an (automatically constructed) thesaurus:

we may retrieve documents that DON'T have the term 'system', but they contain almost everything else ('data', 'retrieval')

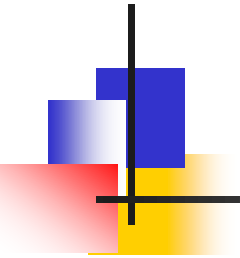
LSI - Detailed outline

- LSI
  - problem definition
  - main idea
  - experiments



LSI – Some experiments

- 150 Tech Memos (TM) / month
- 34 users submitted ‘profiles’ (6-66 words per profile)
- 100-300 concepts



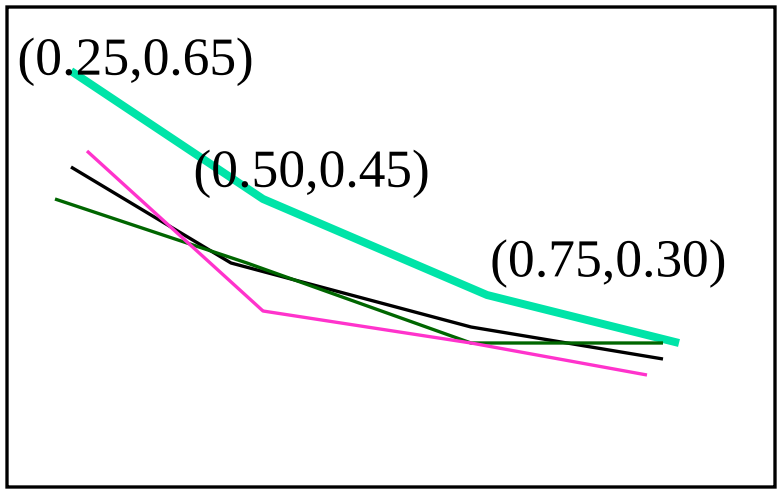
LSI - Experiments

- four methods, cross-product of:
  - vector-space or LSI, for similarity scoring
  - keywords or document-sample, for profile specification
- measured: precision/recall

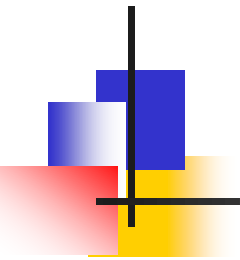
LSI - Experiments

- LSI, with document-based profiles, were better

precision

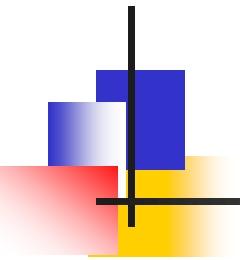


recall



LSI - Discussion - Conclusions

- Great idea,
  - to derive ‘concepts’ from documents
  - to build a ‘statistical thesaurus’ automatically
  - to reduce dimensionality
- Often leads to better precision/recall
- but:
  - Needs ‘training’ set of documents
  - ‘concept’ vectors are not sparse anymore

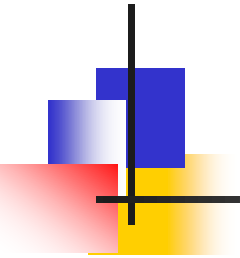


LSI - Discussion - Conclusions

Observations

- Bellcore (-> Telcordia) has a patent
- used for multi-lingual retrieval

How exactly SVD works?



Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
- spatial access methods
- fractals
- text
- SVD: a powerful tool
- multimedia
- ...

References

- Foltz, P. W. and S. T. Dumais (Dec. 1992). "Personalized Information Delivery: An Analysis of Information Filtering Methods." Comm. of ACM (CACM) 35(12): 51-60.