16 The Information Retrieval "Data Model"

16.1 The general model

16.2 Similarity

16.3 Boolean Model

16.4 Vector space Model

16.5 Implementation issues

16.6 Evaluation of Retrieval effectiveness

16.7 IR and DBS

Not presented in the course!
Not relevant for exam.

Lit: Manning, Schütze, Prabhakar: Introduction to Information Retrieval, Cambridge University Press, 2008 (online book for free!)

16.1 The Information Retrieval: the general model

Document model

Document d_n : List of words

$(\mathbf{t}_{1(n)}, \mathbf{t}_{2(n)}, \ldots, \mathbf{t}_{n(d)})$

```
myTxt=[This,is,a,dumb,example]
```

Operation: **find documents matching query**

```
q = "dumb"
```

Less frequent: `insert new document d`

Never: `update d set keyword 'dump'='clever'`

IR Database

D = "set of m documents"

Web pages, scientific papers, newspaper articles...

Problem: Many different words in documents;

Needed: **canonical** document **representation**

Can it be implemented with tables?

Information Retrieval Model

Canonical Representation

Index terms

$K = (k_1,...,k_n)$:

$\sim$ vector of all n words occurring in the database

(A,...ABEL,.... , DUMP,..,DULL, .., IS, .., STUPID,.. TEXT,...)

- Typically **<u>very</u>** large vector.
- Words may be **normalized**
  (TEXT, TEXTS, STUPID, STUPIDITY) ... or not.
- **Stop words** may be eliminated ... or not.
  (be, have, he, she, it, and, ...)

Significance of terms

Freie Universität Berlin

For every dj $\in$ D, $k_i \in$ K there is a **weight** $\boldsymbol{w_{i,j} \geq 0}$,

- $wij \in \boldsymbol{R}$
- **k_i does not occur in d_j =>** $\boldsymbol{w_{i,j} = 0}$

Most simple case:

wij = 1 if term ki occurs in document dj;
otherwise = 0

$\Rightarrow$ document dj represented as a **0/1 vector .**

Sparse vector: most words "of the lexicon" do not occur in a document.

Naive representation:Term / document matrix

						(6)	(7)	(8)	(9)
aber						0	0	0	1
auch						0	1	0	0
bein						0	0	1	0
da						0	0	1	1
dicht						1	0	0	0
gaul						0	0	1	0
goeth						1	0	0	0
hinschmetzen	0	0	0	0	0	1	0	0	0
ich									
kafka									
kannst									
kant									
knecht									
mann									
pferd	0	0	0	0	0	0	0	1	0

Herr und Knecht
Der Herr rief: lieber Knecht
"Mir ist entsetzlich schlecht!"
Da sprach der Knecht zum Herrn:
"Das hört man aber gern"

herr und knecht der herr rief lieber knecht mir ist entsetzlich schlecht da sprich der knecht zum herr das hör man aber gern

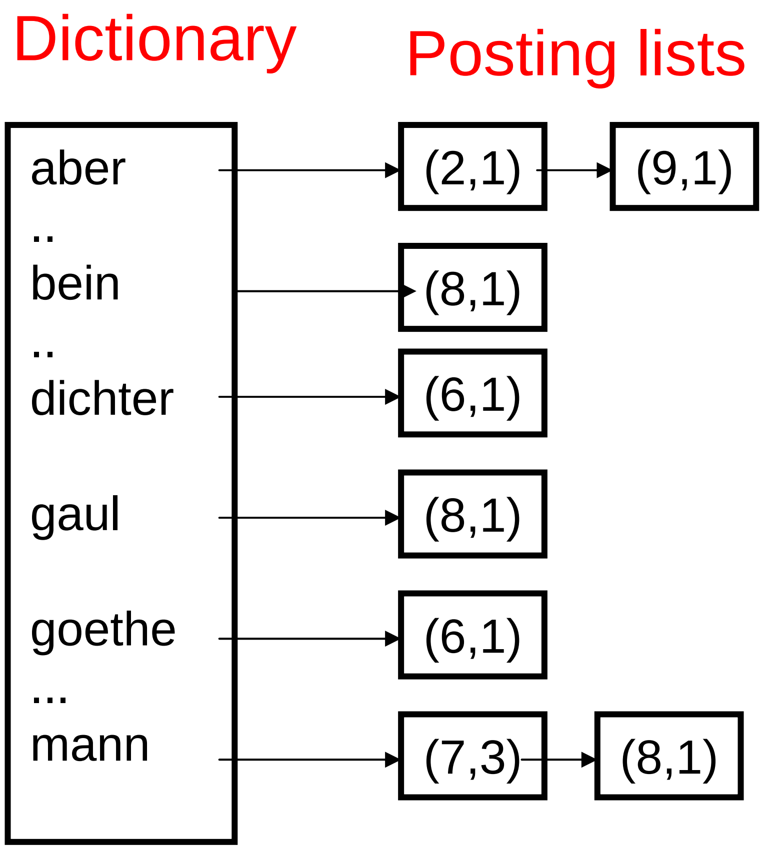
What is a query?
Boolean expression of keywords,
not SQL

not "goethe" AND "dichter" OR "mann"

→ Result is: {(7), (8)}

Posting list representation

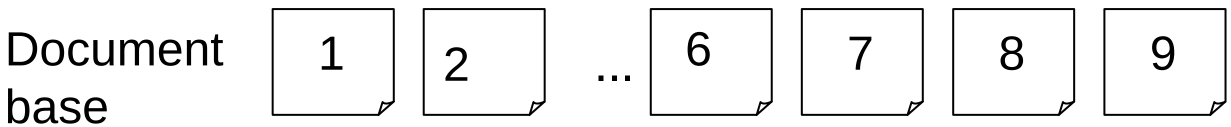
Freie Universität Berlin



Posting list entry:
`(doc#, "weight")`

$w_{ij} \in \{0,1\}$ means:
t_i occurs in d_j (1) or not (0)

$w_{ij} = 0$ represented by posting list



Information Retrieval model

Freie Universität Berlin

Queries

What does "***query q <u>matches</u> document d***" mean?

Different from SQL etc.,

Why?

Principle problem:

No agreement about semantics of d and q:
Would require formal understanding of Natural Language

The best we can do now:

Formalize **similarity** between documents and query

Information Retrieval: the problem

Given:

- **Document set D**= {d1,…dm}
- **Query q**

Find an order $di_1 \geq di_2 \geq \ldots \geq di_m$ of D

such that:

$\boldsymbol{sim\,(di_k, q) \geq sim\,(di_{k+1}, q)\,,\ k = 1..m-1}$

for some **similarity measure** *sim*.

IR query processing :

find an order (**ranking**) of all documents, such that the rank of a document conforms to its similarity with the query q.

16.2 Similarity and distance

Measuring

A metric in some space S is a real valued function
$m: S \times S \rightarrow \mathbf{R}$
with properties:

$m(x,y) = 0 \Leftrightarrow x=y$
$m(x,y) \geq 0$ for all x,y
$m(x,y) = m(y,x)$ for all x,y
$m(x,y) + m(y,z) \geq m(x,z)$ for all x,y,z

Distance and similarity

Distance measure of x,y in a space S:

Metric function, which assigns real number to {x,y}

Well known examples:

Euclidean distance in $\mathbf{R}^n$ $\quad \sqrt{\Sigma\,(xi-yi)^2}$

Minkowski metric $\quad \sqrt[s]{\Sigma_s\,(xi-yi)^s}$

s-> ∞: **Maximum metric** $\quad \max |xi-yi|$

s= 1 **Hamming distance** $\quad \Sigma\,|xi-yi|$

(Manhatten block distance)

16.3 The Boolean retrieval model

Documents: $w_{ij} \in \{0,1\} \Rightarrow dj \in \{0,1\}^n$

Query language:

Boolean expression of $k_i \in K$

Semantics of query q:

let dj $\in$ D be a document vector of 0 and 1,

q = k_i then d matches q iff $d_{ij} = 1$

q = "q1 AND qj "

q matches d_{ij} if q1 matches d_{ij} *and* q_2 matches d_{ij}

q = "q_1 OR q_j "

q matches d_{ij} if q1 matches d_{ij} *or* q2 matches d_{ij}

q = "NOT q1"

q matches d_{ij} if q_1 does *not* match d_{ij}

Boolean Retrieval

Similarity?

Not well defined, since **queries and documents have different representation.**

Binary decision: d matches q or not

Implementation

Conceptually simple

Efficient query evaluation

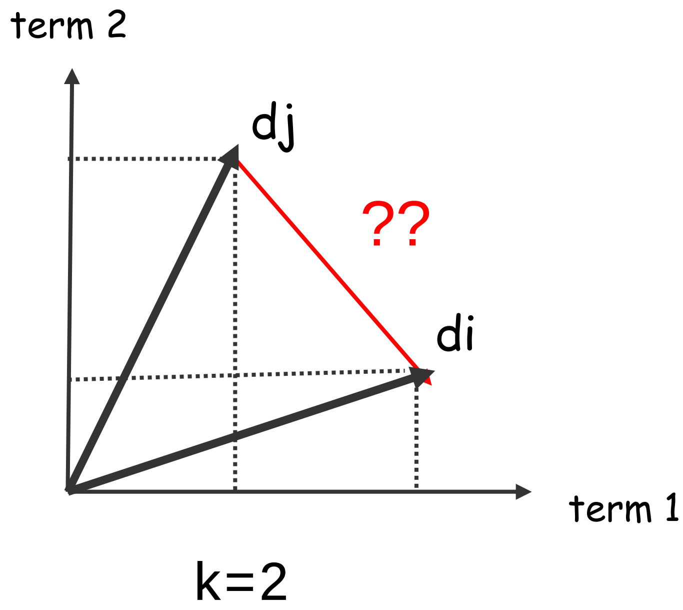
Several Boolean IR system still operational

16.4 Vector space model

Model

Documents: **points in a |K| = n – dimensional vector space.**

- **Weights** normalized
  e.g. 0 <= w <= 1
- **Terms** are **independent** of each other ("orthogonal")



Queries …..

…. are (formally) **documents**: q= (q1, q2, …,qn)

Measure of similarity between document and query? vector difference?

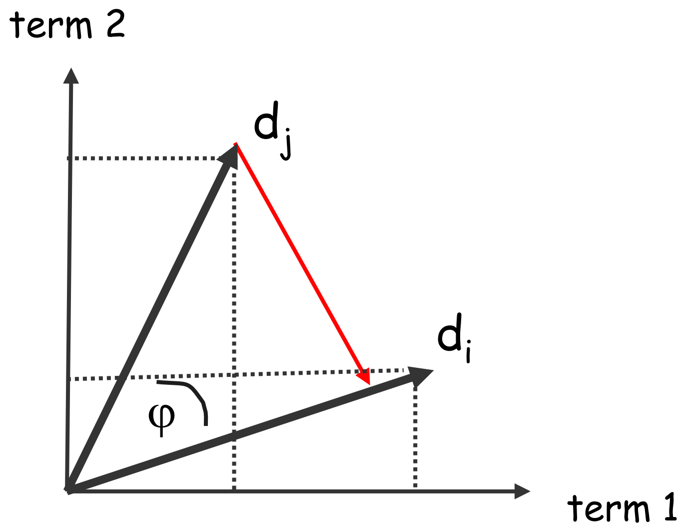
Vector space: similarity function

Heuristic similarity functions

Scalar product?

$w_{1j}*q_1 + w_{2j}*q_2 + ... + w_{nj}*q_n$

not bounded, may become arbitrarily large



Cosine measure

$\mathrm{Cos}\,(d_j,q) = \cos\varphi$

$= d_j \bullet q \;/\; |d_j| * |q|$

$= \sum w_{ij}*q_i \,/\, \sqrt{(\sum w_{ij}{}^2)} * \sqrt{(\sum q_i{}^2)}$

= normalized scalar product

Measures angle between query vector and document.

Weights

How to assign weights to documents / queries

Manual weight? Impossible! (more or less)

Document frequency

Infrequent terms are typically more significant than frequent ones

"system" compared to "transaction"

Justification:

Zipf's law

"Frequency f of an event is inversely proportional to its rank r " or r*f $\approx$ const

(Human Behaviour and the Principle of Least effort (G. Zipf 1949))

Zipf law, example

Freie Universität Berlin

Das zipfsche Gesetz am Beispiel des Brown- und des LOB-Korpus

Rang	Anzahl	R*A/100000	Term
1	138323	1,3832	the
2	72159	1,4432	of
3	56750	1,7025	and
4	52941	2,1176	to
5	46523	2,3262	a
6	42603	2,5562	in
7	22177	1,5524	that
8	21210	1,6968	is
9	20501	1,8451	was
10	19587	1,9587	it
100	2043	2,0430	years
500	394	1,9700	program
1000	207	2,0700	jones
2000	105	2,1000	granted
3000	67	2,0100	agencies
4000	47	1,8800	embassy
5000	36	1,8000	vale
10000	14	1,4000	poisoning
12034	11	1,3237	yell

"Less frequent terms are more informative"

Consistent with information theory of C. Shannon

From R. Ferber, Information retrieval

Document frequency

First hypothesis: **importance of a term depends on number of documents it occurs in**

=> Weight w of term t should be **inverse proportional to document frequency DF**

Document frequency df_i
is the number of documents (of some collection) term t_i occurs in.
Inverse document frequency (IDF): $1/df_i$

Weights

Second hypothesis:

A term t_j occuring more often in a document dj than t_k characterizes d_j better than t_k

Term frequency

Term frequency f_{ij} of term t_i and document d_j is defined as the number of occurences of t_i in d_j

Example: if the term 'database' occurs many times in a paper, but 'processor' only once, then the paper is more likely on databases than on processors.

Note: "Semantics" is approximated statistically

Weight normalization

Observation

- **Length of a document** influences number of occurrence of a word
- Document frequency i**ncreases with number of documents** in database

$\Rightarrow$ **Normalization heuristics**

TF normalization heuristics : e.g. $r_{ij} = \log(1+f_{ij})$

IDF normalization heuristics:

e.g. weight of term i: $w_i = \log(1 + m / df_i)$

Many heuristics analyzed over the years, above one appropriate in many cases

Number of documents in collection

Cumulative Weight

Freie Universität Berlin

Weight of term t_i in document d_j ?

$$w_{ij} = f\,(\,TF,\,1/DF) \;=\; f\,(TF,\,IDF)$$

"TF / IDF heuristic":

$w_{ij} = r_{ij} * w_i$ is the weight of term t_i in a particular document collection,

$r_{ij} = log\,(1+f_{ij}),\; w_i = log\,(1 + m\,/\,df_i)$

$\Rightarrow$ Weight of a query term

$w_{jq} = q_t * w_j$, ,

q_j = weight of j-th query term in query q

Typical: $\mathbf{q_j = 1}$ ("All query terms equally important")

Ranking using the cosine measure

$$Cos(d_j, q) = d_j \bullet q \;/\; |d_j| * |q|$$

$$= \sum_{t_i \in d_j \cap q} w_{ij} * w_q \;/\; \sqrt{(\sum w_{ij}^{\ 2})} * \sqrt{(\sum w_{iq}^{\ 2})}$$

$t_i \in d_j \cap q$ (terms in query *and* document)

$$= 1/(W_j * W_q) \;*\; \sum \log(1 + f_{i,j}) * (\log(1 + m/df_t))^2$$

where W_i and W_q are the normalization constants (i.e. length of documents / query vector)

quadratic term influence when using $\mathbf{w_q}$

Most often used **cosinus measure**:

$$\boldsymbol{simCos(d_j, q) = 1/(W_j * W_q) \;*\; \sum_{t \in d_j \cap q} \log(1 + f_{tj}) * \log(1 + m/df_t)}$$

using **weight 1 for** all **query terms**

16.5 Cosine measure: implementation

Ranking of document set with respect to q:

Calculate cos(d,q) for each document d $\in$ D

Efficiency?

$log(1+ m/df_t)$ ~ **constant** c_t for term t

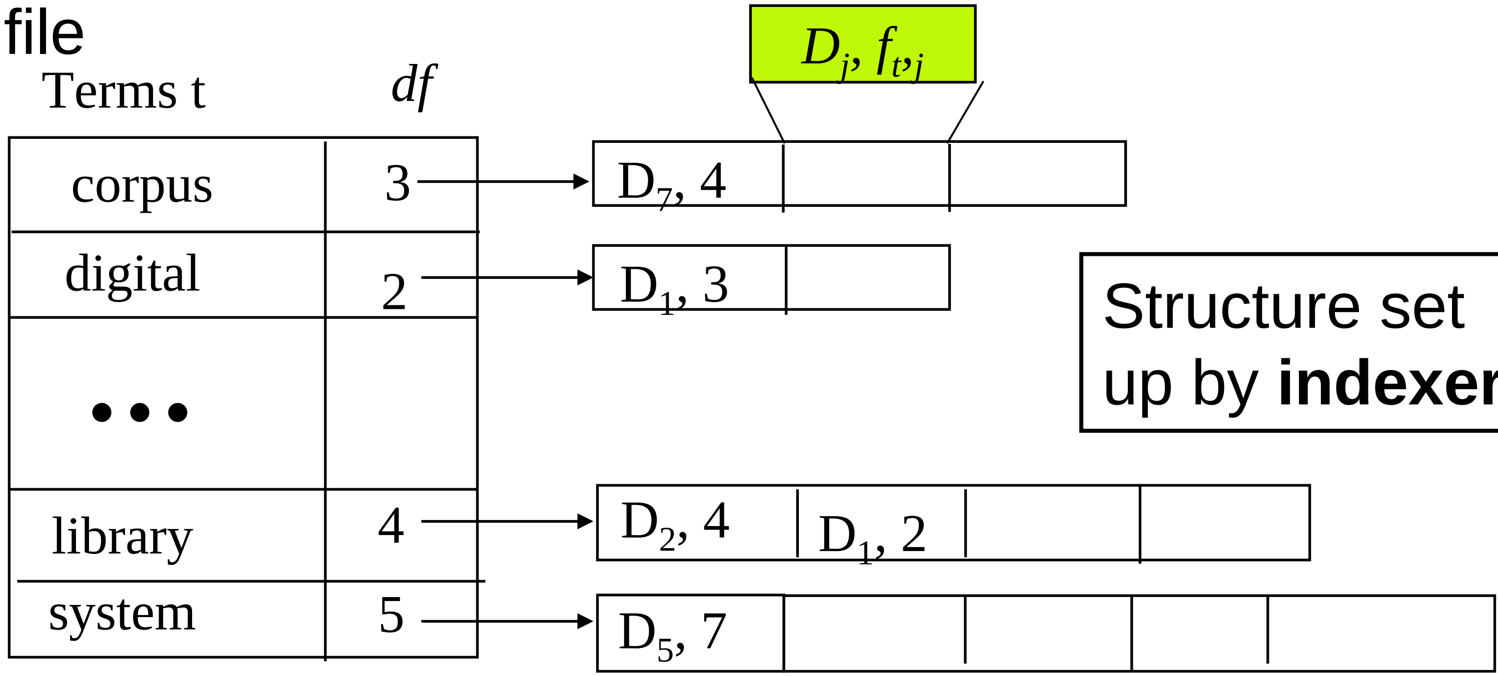
$1+log\ f_{ij}$ is a **constant** c_{ij} for each term / document pair

$\Rightarrow$ store $c_t * c_{ij}$ with each term / document pair

$\Rightarrow$ **sim (d_j, q) is calculated as a sum of stored constants**

Implementation of vector space model

Inverted file



Structure set up by **indexer**

Index file,
e.g. B+ tree,
suffix tree

Postings lists,
suited for boolean queries
and similarity search

Most values may be calculated in advance (see above) and put into the posting list, many refinements, e.g. position of term in text allows phrase search!

Top-k

Freie Universität Berlin

- How can the k documents most similar to q be found without ranking all documents?

  Google output:
  Ergebnisse **1** - **10** von ungefähr **44.700.000** für **information retrieval**. (**0,15** Sekunden)

- Similar problem also in databases:

  SELECT top-5 of (title, year, ...)
  FROM Movies m, Actor a, ...
  WHERE m.title LIKE... AND a.name = ... AND...

  Which movies match best? How to find them efficiently?

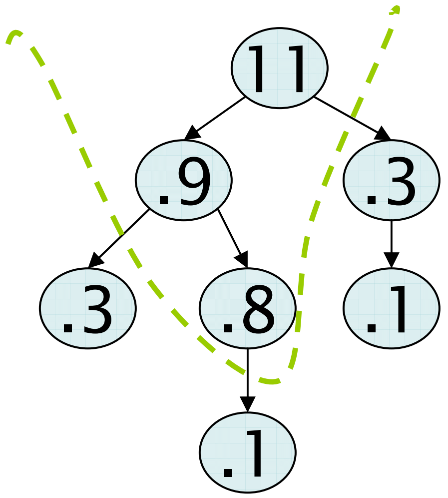
Use heap for selecting top *k*

Freie Universität Berlin

Binary tree in which each node's value > values of children

Takes $2n$ operations to construct, then each of $k \log n$ "winners" read off in 2log *n* steps.

For *n*=1M, *k*=100, this is about 10% of the cost of sorting.



Ranking: many heuristics

Freie Universität Berlin

Web documents may be ranked independent of queries:

Page rank: measures "importance" of a page dependent on link structure (in the Web)

$R(p) = (1\text{-}d) + d^* \Sigma R(s_j)/c_j$

$R(p)$ rank of page p

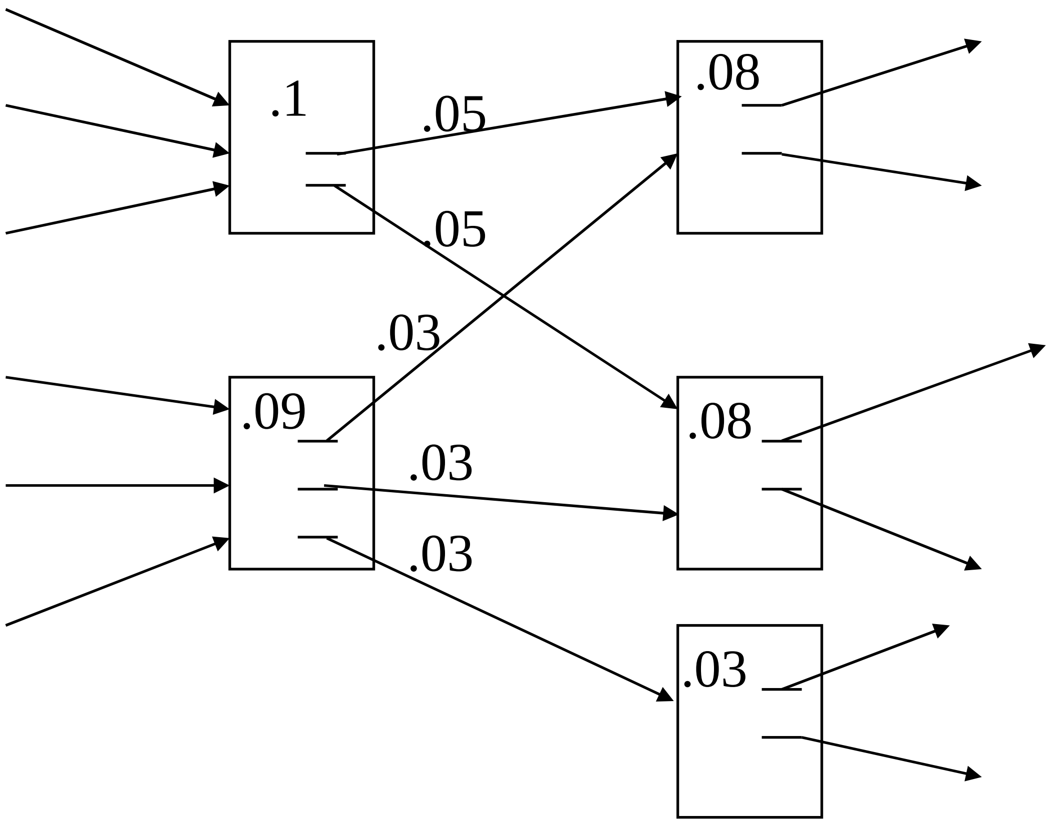
s_j : page j with link to p, $R(s_j)$: rank of page s_j

c_j: number of pages s_j has links to

Google etc use vector space metrics, page rank and many more heuristics

Initial PageRank Idea (cont.)

Can view it as a process of PageRank "flowing" from pages to the pages they cite.



Initial PageRank Idea

Freie Universität Berlin

- Just measuring in-degree (citation count) doesn't account for the authority of the source of a link.
- Initial page rank equation for page *p*:

$$R(p) = c \sum_{q: q \rightarrow p} \frac{R(q)}{N_q}$$

N_q is the total number of out-links from page *q*.

A page, *q*, "gives" an equal fraction of its authority to all the pages it points to (e.g. *p*).

c is a normalizing constant set so that the rank of all pages always sums to 1.

Initial Algorithm

Iterate rank-flowing process until convergence:

Let *S* be the total set of pages.

Initialize $\forall p \in S$: $R(p) = 1/|S|$

Until ranks do not change (much) (*convergence*)

For each $p \in S$:

$$R'(p) = \sum_{q:q \rightarrow p} \frac{R(q)}{N_q}$$

For each $p \in S$: $R(p) = cR'(p)$ (*normalize*)

$$c = 1 / \sum_{p \in S} R'(p)$$

Google Ranking

- Complete Google ranking includes (based on university publications prior to commercialization).
  - **Vector-space similarity component.**
  - **Keyword proximity component.**
  - **HTML-tag weight component (e.g. title preference).**
  - **Pagerank component.**
- Details of current commercial ranking functions are trade secrets.
- Many variations, e.g. personalization, modify jumping to random page ("teleportation"),
  e.g. if I am soccer fan, I will more often jump to soccer pages, even if there is no link.

16.6 Evaluation of query results

Issues

Subjectiveness of judgement

How relevant is a document with respect to a query?

Elaborate, costly **empirical tests** required

many queries, many individual judgements for each query, mean value of judgements?

Evaluation model

Ideal observer: knows relevant documents for each query

Check for each query q

- how many relevant documents found
- how many irrelevant documents found

Calculate mean over many queries

Evaluation

	relevant	not relevant
found	r	n
not found	v	u

False negative False positive

Recall:

fraction of *relevant* documents *found* of all *relevant* documents

$$R = r / (r+v)$$

Precision:

fraction of *relevant* documents *found* in the set of documents *found*

$$P = r /(r + n)$$

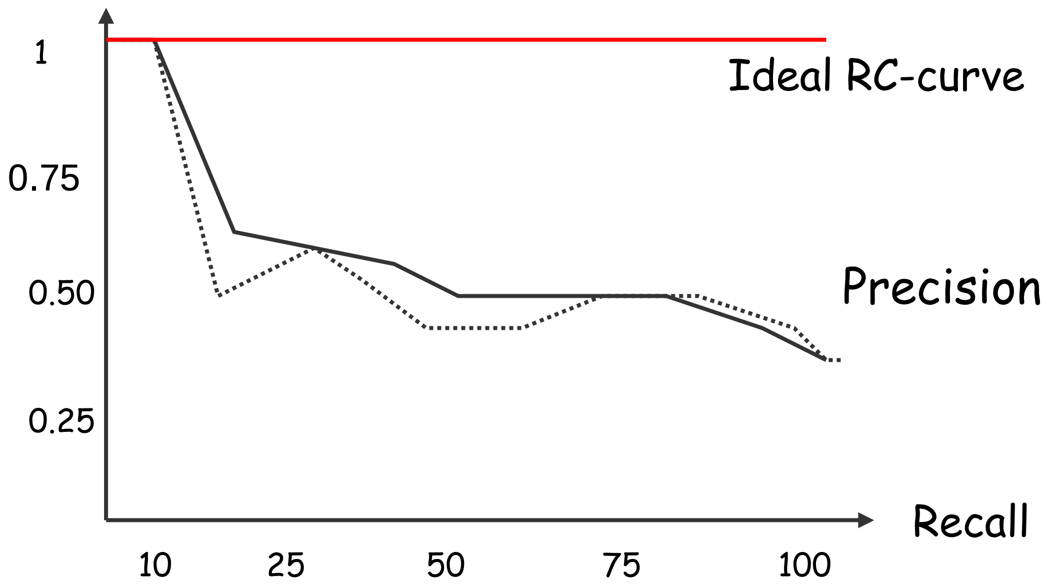
How to evaluate ranking *order*?

F-measure:

$$F = 2*R*P/(R+P)$$

Recall-Precision Graph

	Recall level	Precision %
1 ✓	10	100
2	10	50
3	10	33
4 ✓	20	50
5 ✓	30	60
6	30	50
7 ✓	40	57
8	40	50
9	40	44
10	40	40
11	40	36
12 ✓	50	42
13 ✓	60	46
14 ✓	70	50
15	70	47
16 ✓	80	50
17	80	47
18	80	44
19 ✓	90	47
20	90	45
21	90	43
22 ✓	100	45
23	100	43
24	100	42
25	100	40



Recall level n:
n % of all relevant Documents have been found

RC curve:
Precision at recall level n

for a particular query and document set!

16.7 Database and Information Retrieval

- High end DBS use **text extenders** for combining (relational) database functionality and retrieval
- Different technical approaches within one system: e.g. **user defined type 'text'** or embedding of a **search engine**
- Different kind of indexes
  - e.g. Posting list,
  - database index for small text snippets,
  - specific index for text classification
- SQL extension for text

IR queries on texts as DB object

Freie Universität Berlin

Querying

– different kinds of query language

- e.g. Boolean queries, simple keyword queries and more

SQL like search predicates (Oracle)

```
CREATE Table MovieTab
AS (id INTEGER, txt CLOB)
 SELECT id, txt, SCORE(1)
   From MovieTab
   WHERE CONTAINS (txt, 'Monroe',1)>0 ;
```

Place holder for relevance

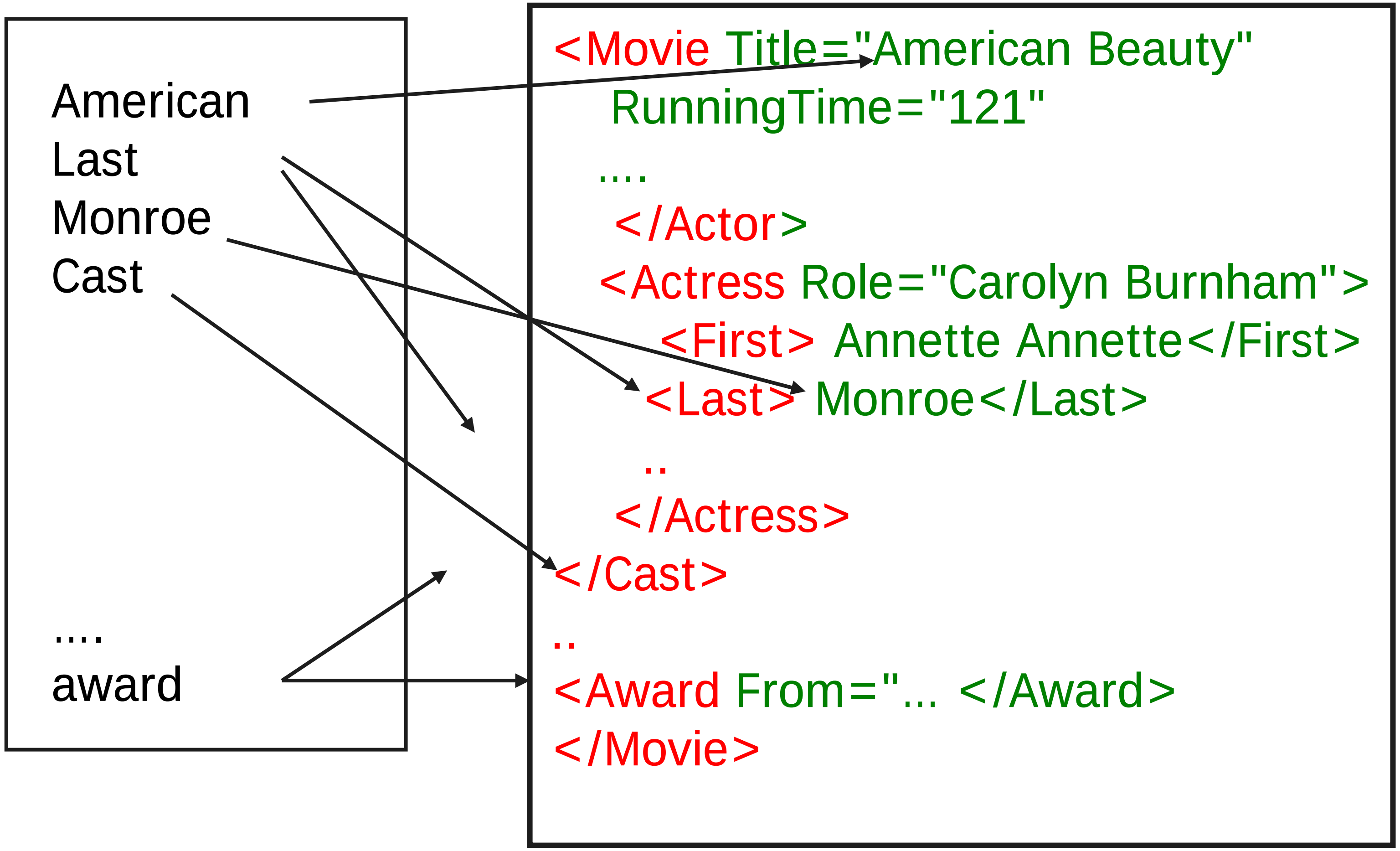
Score: relevance measure

Value of CONTAINS: 0..100

Text (XML) document as a DB object

Freie Universität Berlin

Indexed 'text ' attribute



Does not only show in which doc a term occurs, but also its position !

Summary

- Information Retrieval deals with **unstructured data,** in particular text,image, time series, sound... more difficult, but important
- **Vector space systems** outperform (and similar models) outperform Boolean retrieval
- **Similarity and ranking** important also in traditional (relational) databases
- Integration of "structured" and "unstructured" data is an important topic. First step was : text in RDB