Query languages 1 (NDBI001)

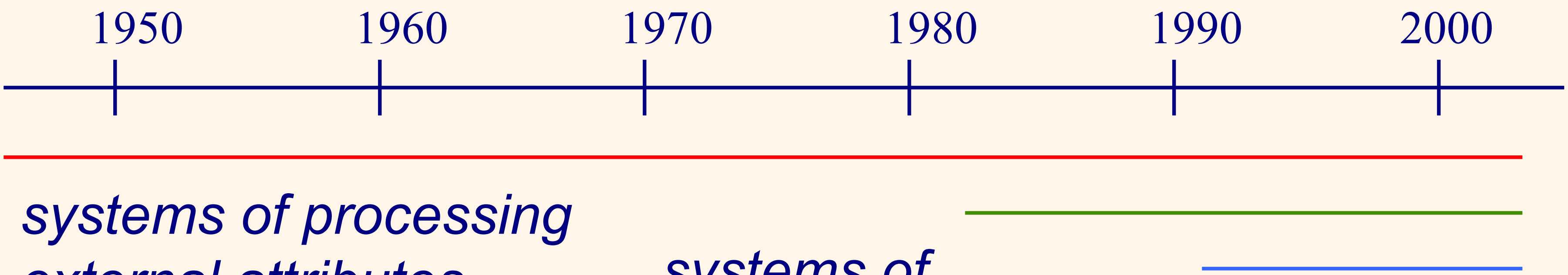
Information retrieval

Jaroslav Pokorný

MFF UK, Praha

pokorny@ksi.mff.cuni.cz

Information retrieval systems - development



systems of processing external attributes

systems of fulltexts processing

digital libraries

Resources:

- creating texts directly in computer
  - a need - searching, not only browsing
  - indexing not always possible
- development of large storages (CD ROM, WORM)
- development of communications (Internet)

Content

1. Introduction
2. Measuring the relevance
3. Boolean model
4. Vector space model
5. Relevance feedback
6. Thesaurus
7. Conclusions

Text retrieval

query - request formulated in a language is given by a text pattern (word, expression, a substring of a word, phrase, or a whole text) or by several patterns (*conjunctive query*)

More generally: Boolean expression

*answer (*set of *hits)* - texts matching a query

hit relevance – the degree to which the hit matches the user request. The notion of relevance is imprecise, context- and user-dependent.

- answer restriction:
  - maximum M
  - maximum M most relevant
  - set a threshold Θ

Text retrieval

Field: **Information Retrieval**

IR is all about retrieval, what you want, when what you want, is hidden in mass of what you do not want.

More precisely: find for a query relevant documents

Field: **Information Filtering**

Assign to a document D profiles in such way, that D is for them relevant.

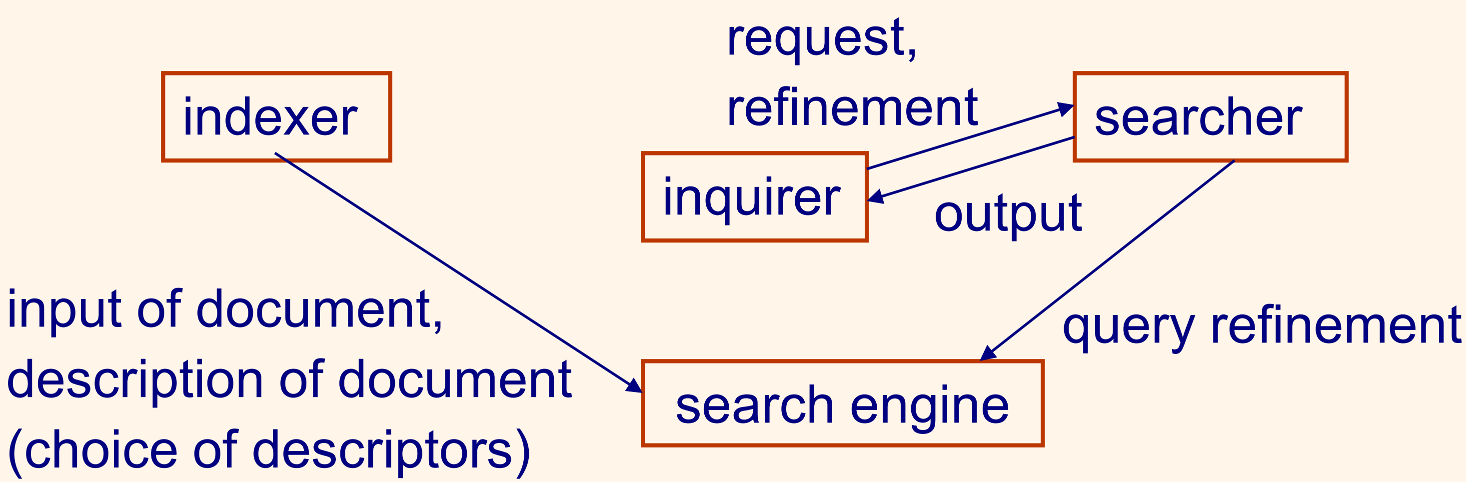
IR - basic architecture

Subsystems: text disclosure (1)

text delivery (2)

(1) see information services

secondary information versus fulltext



historical model

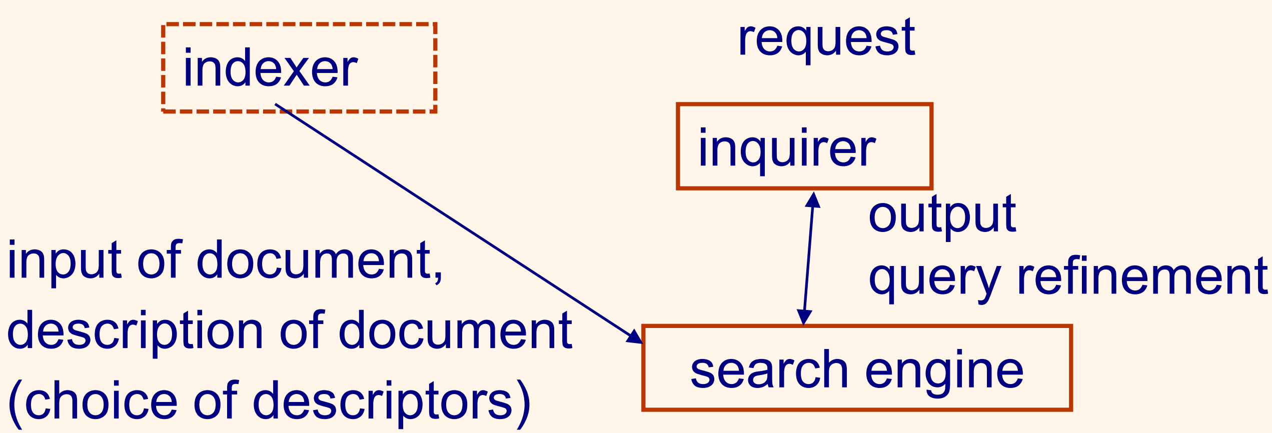
IR - basic architecture

Subsystems: text disclosure (1)

text delivery (2)

(1) see information services

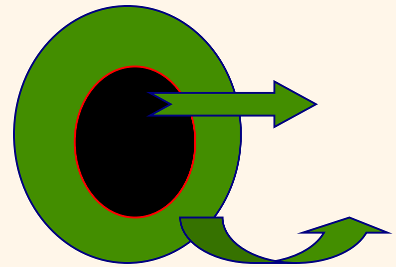
secondary information versus fulltext



current model

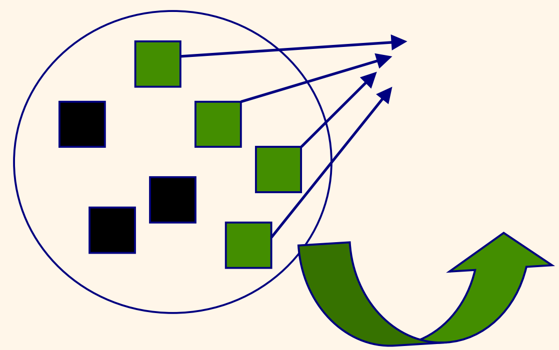
Measuring the relevance

Recall R



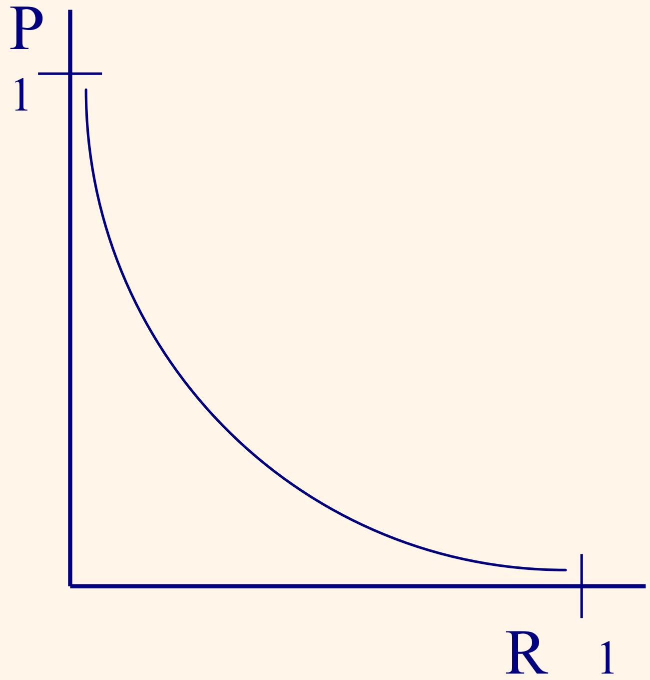
$$R = \frac{\text{\#retrieved relevant documents}}{\text{\#relevant documents in collection}}$$

Precision P



$$P = \frac{\text{\#retrieved relevant documents}}{\text{\#retrieved documents}}$$

Trade-off between R and P



precision-recall curve

Boolean model

- Document representation: a set of terms
- Querying:
  - formally: by Boolean expressions
  - technique: exact match
- Determining terms - practice:
  - removal of stop-words from sets of terms
    result: reduction 30-50% (C.J. van Rijsbergen)
  - linguistic processing (tokenization)

Boolean model

One of possible syntaxes:

<term>

<attribute_name> = <attribute_value> /comparison/

<function_name>(<term>), /function application /

X AND Y	retrieve D, containing both X and Y.
X OR Y	retrieve D, containing either X or Y.
X XOR Y	retrieve D, containing either X or Y but not X AND Y
NOT Y	retrieve D, not containing Y
X adj Y	retrieve D, that contain X followed by Y
X (n)words Y	retrieve D, that contain X followed by Y at the maximum distance *n* words
X sentence Y	retrieve D, in which X and Y occur in the same sentence

Boolean model

- . will match arbitrary character.
- * character followed by * will match arbitrary number of occurrences (including 0) of this character. E.g., xy* will match x, xy, xyy etc.
- + character followed by + will match arbitrary number of occurrences (except empty) of this character. E.g., xy+ will match xy, xyy, xyyy etc.
- [] characters in [] will match arbitrary one character, which is in brackets, but not another. E.g., [xyz] will match x, y or z.
- [^] starting the string in [] by ^ means negation (not). E.g., [^xyz] will match arbitrary character except of x, y, or z.
- [-] - among characters in [] denotes a range of characters. E.g., [a-x] will match arbitrary character from a to x.

Boolean model: P versus R

- By query refinement in Boolean model we can obtain higher P, but lower R.

Ex.: experiment (Blair, Maron,1985) - 40000 legal texts

Goal: not only high P, but R as well.

Results: P $\rightarrow$ 80%, R $\rightarrow$ 20%

- the synonym problem – too general language, it is not possible to capture it by thesaurus.

Ex.: accident, disaster, collision, „something happened“, …

- automatic indexing does not eliminate these problems

Boolean model: problems

What affects the relationship P and R?

Problems with manual indexing:

indeterminacy

- in indexing *influence of indexer*
- in selection of terms for query *influence of inquirer*

Ex.: p_1, p_2 probabilities, that the inquirer uses terms t_1, t_2

q_1, q_2 probabilities, that the terms t_1, t_2 se vyskytují in D

$\Rightarrow$ p, that the inquirer selects t_1, t_2 and D with t_1, t_2 is retrieved, is

$$p_1 * p_2 * q_1 * q_2$$

E.g., R = 0,6 * 0,7 * 0,5 * 0,6 = 0,126 $\Rightarrow$ R < 13%

$\Rightarrow$ for i=5, $p_i = q_i = 0{,}5 \Rightarrow R = 0{,}1\%$

$\Rightarrow$ if there is 1000 relevant D, only 1 is retrieved!

Boolean model: problems

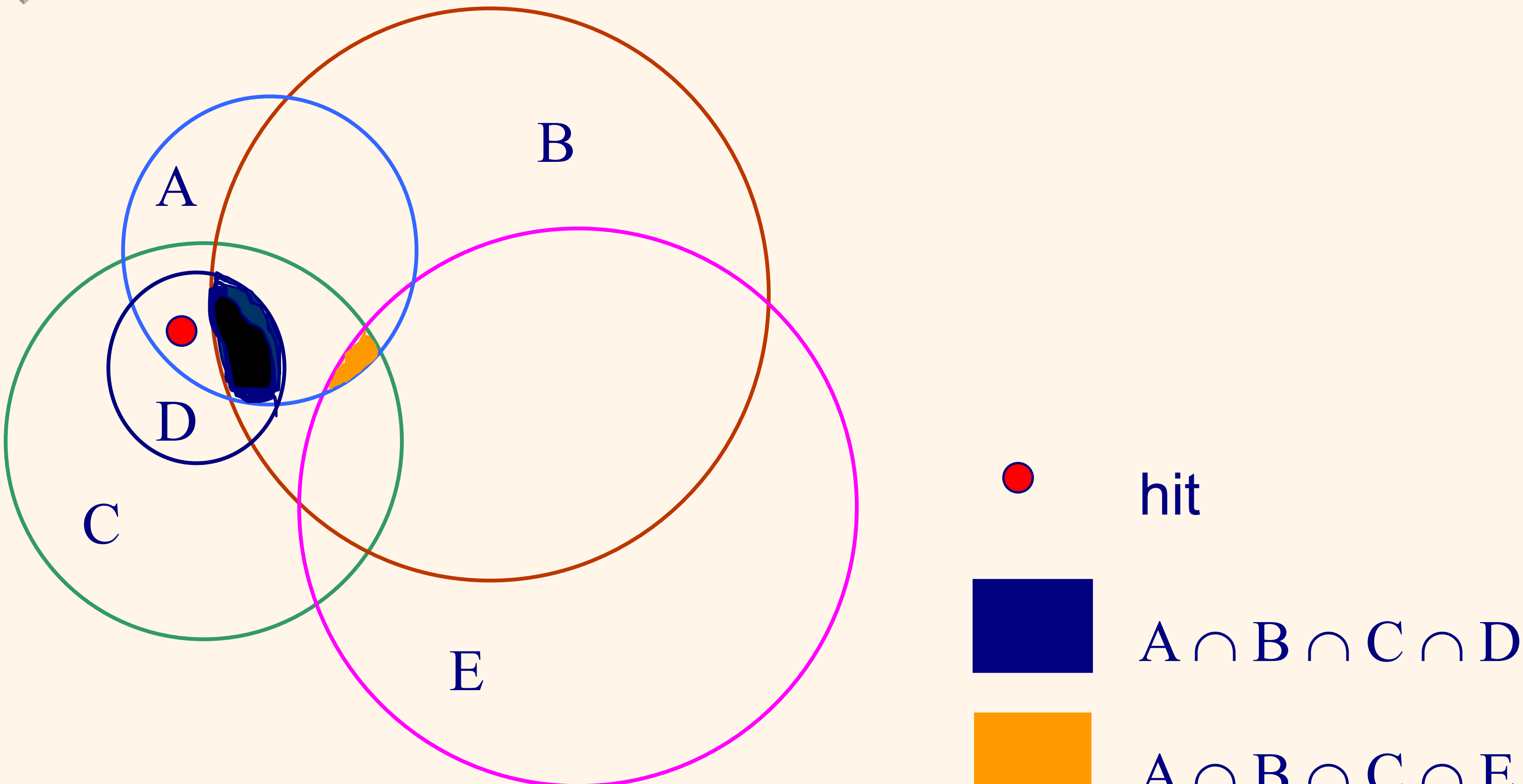
prediction criterion – how to ensure a match between selection of terms for query and for documents (today: similarity of ontologies)

- method: elimination of indeterminacy

maximum criterion – to handle up to 20-50 hits

- problems with fulltext collections:
  - *collection size* (versus maximum criterion)
  - *selection of terms for query*
    - revaluation of elimination of indexers
    - indeterminacy of inquirer remains
  - *unilateral behavior of inquirer* -
    tendency to change the last decision and retain the first steps

Boolean model: problems



Boolean model: problems

Indeterminacy of the inquirer's selection of search terms

Solution:

- lookup D with high relevance for inquirer (D is known + it is known, that it occurs on collection),
- terms for query are retrieved from D,
- omitting terms resp. replacing them by disjunctions.

$\Rightarrow$ decreasing the inquirer indeterminacy

Boolean model: problems

Solution of unilateral behavior of inquirer by weighting:

Ex.:

	terms		*probability (weight)*
Author: Pokorný			0,3
Date: 1995-1999			0,7
Journals:	CW		0,2
	Artificial Intelligence		0,5
	ERCIM News		0,2
Descriptors:		XML	0,6
		database	0,8
		query language	0,9

The total number of conjunctive queries is 255.

Boolean model: problems

Products of probabilities for

2 terms	3 terms	max. for 1, 2, ...
$p_{do} * p_{da} = 0{,}72$	$p_{do} * p_{da} * p_{dat} = 0{,}5$	0,9
$p_{do} * p_{dat} = 0{,}63$	$p_{do} * p_{dat} * p_{xm} = 0{,}38$	0,72
$p_{da} * p_{dat} = 0{,}56$	$p_{do} * p_{da} * p_{ar} = 0{,}4$	0,5
…	…	0,3
		0,15

Algorithm:
- create groups for all combinations
- calculate maxima for groups
- is the maximum criterion met?
- offer to the inquirer

Boolean model: other problems

- Non-intuitive results
  - A AND B AND C AND D AND E

    D not containing only one of given terms will be not retrieved.
  - A OR B OR C OR D OR E

    D containing only one from given terms are seen as equally important as documents containing all given terms.
- It does not allow output size control.
- all Ds satisfying a query are conceived as equally important, it is not possible to sort them by their similarity.

Boolean model: other problems

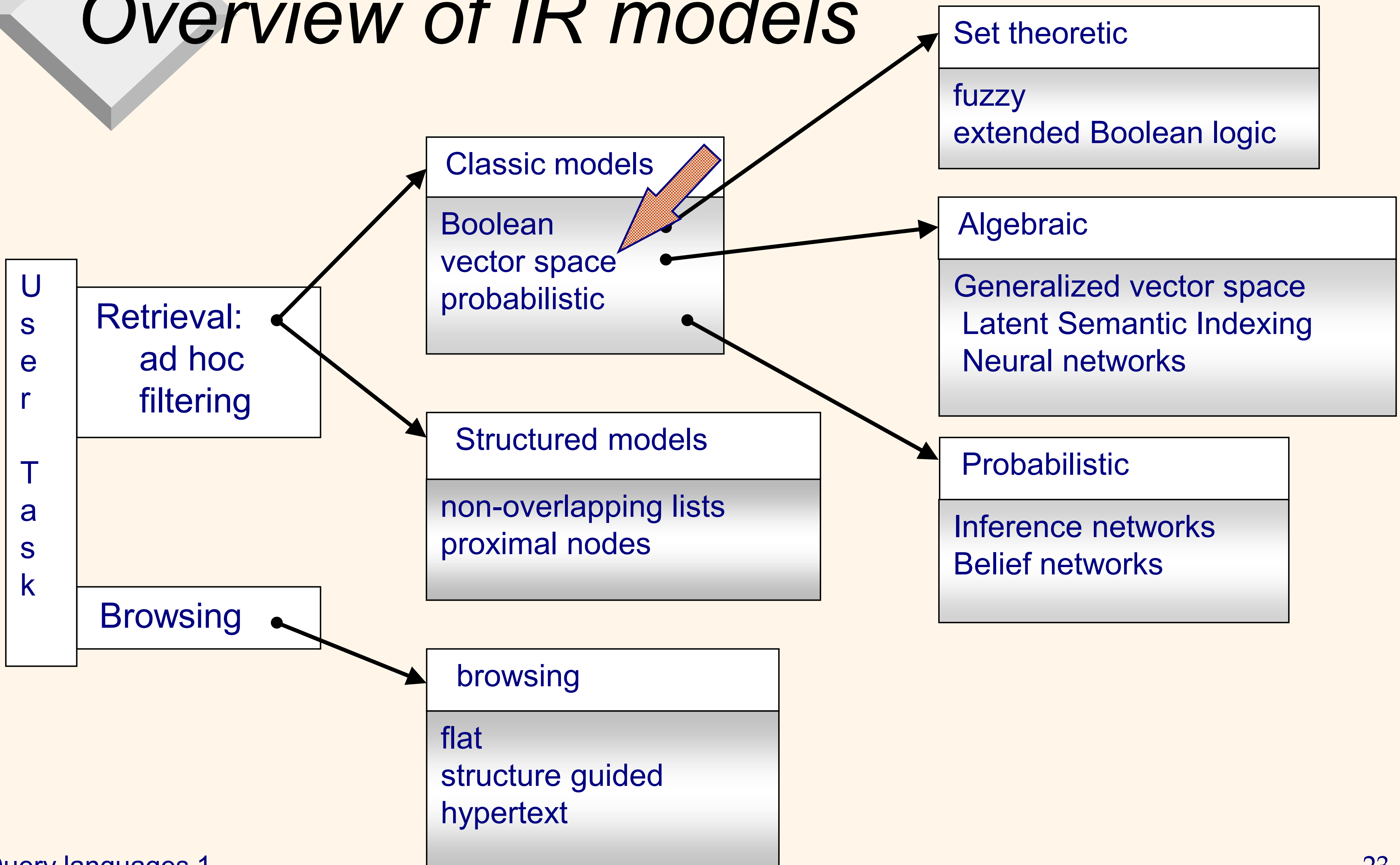
- It is difficult to realize automatic relevance feedback, i.e. to modify automatically a query based on D marked in answer as relevant.
- Expressive power of Boolean model is restricted. Any set {D} of documents describable by terms, can be, in principle, retrieved by an appropriate Boolean query. However, in practice it is not guaranteed for any set {D} satisfying user's needs, to formulate simply Boolean query.
- more art than science.

What next?

Thesis:

Classic Boolean systems can be extended by a function influencing maximum criterion; however, it is not possible to increase P and R simultaneously without additional information.

Overview of IR models



Vector space model

Assumption: collection **D** of *m* documents, *n* different terms $t_1 \ldots t_n$

Each document $D_i \in$ **D** is represented by a vector

$$D_i = (w_{i1}, w_{i2}, \ldots, w_{in}), \text{ where } w_{ij} \in \langle 0;1 \rangle$$

where w_{ij} is the weight of a term t_j for document D_i.

D is representable by term-document matrix

$$\mathbf{D} = \begin{matrix} w_{11} & w_{12} & \ldots & w_{1n} \\ w_{21} & w_{22} & \ldots & w_{2n} \\ \ldots & & & \\ \ldots & & & \\ w_{m1} & w_{m2} & \ldots & w_{mn} \end{matrix}$$

Vector space model

- Querying: we regard query as a short document
  - formally: by a query vector
  - partial match querying

    technique: by a similarity function (coefficient)

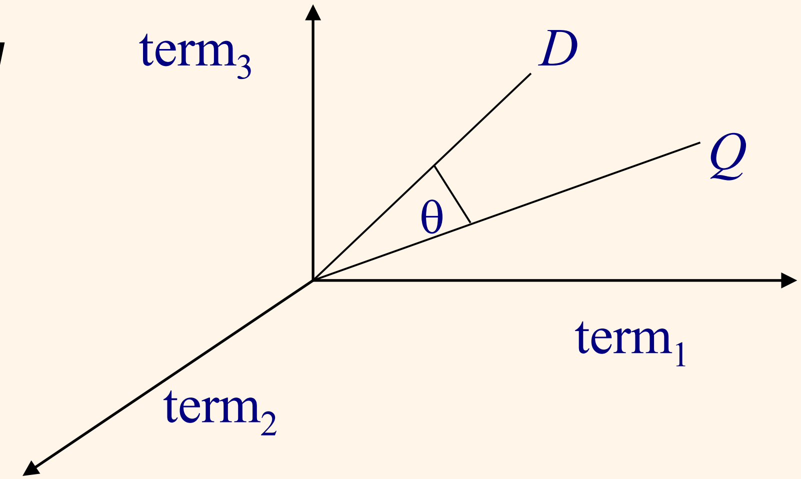
query expression Q in vector model

$$Q = (q_1, q_2, ..., q_n), \text{ where } q_j \in \langle 0;1 \rangle.$$

Problem: how to calculate similarity

- It is possible to rank the retrieved documents in the order of presumed relevance.
- It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

Vector space model



Angle versus distance

- Why not a distance?
- Experiment: we take a document D and append it to itself. The document D′ will be created.
  - “Semantically” D and D′ have the same content.
  - Euclidean distance in the space between points D and D′ would be large.
  - The angle between D and D′ (as vectors) is 0, which corresponds to maximal similarity.
- Key idea: rank documents according to angle between D and query vector.
- Appropriate: cosine – monotonically decreasing function for the interval [0°, 180°]

Vector space model

coefficient similarity (angl. *similarity*) query Q and document D_i

(a) $Sim(Q,D_i) = \sum_{k=1,\ldots,n}(q_k * w_{ik})$ (*dot product*)

(b) $Sim(Q,D_i) = \sum_{k=1,\ldots,n}(q_k * w_{ik})/\sqrt{(\sum_{k=1,\ldots,n}(w_{ik})^2 * \sum_{k=1,\ldots,n}(q_k)^2)}$ (*cosine measure*)

Denominator in (b) is a *normalization factor*,

(c) $Sim(Q,D_i) = 2\sum_{k=1,\ldots,n}(q_k * w_{ik})/(\sum_{k=1,\ldots,n}(w_{ik})^2 + \sum_{k=1,\ldots,n}(q_k)^2)$ (*Dice coefficient*)

Postulate: the more two vectors that represent documents are „near“, the more the documents are similar

Vector space model

Remark: binary vector space model (i.e., the only non-zero w_{ik} in D_i and Q are equal to 1).

For all three cases *Sim* =

- $|Q \cap D_i|$
- $(|Q \cap D_i|)(\surd|Q| * \surd|D_i|)$
- $2(|Q \cap D_i|)(|Q| + |D_i|)$

Advantage: *R* and *P* can be increased up to 20%.

Vector space model

Pragmatic approach: one-word terms + appropriate method of weighting

Term Frequency

TF_{ij} the frequency of t_j in D_i (the number of times that t_j occurs in D_i.

Normalized Term Frequency

NTF_{ij} the frequency of t_j in D_i given as $((TF_{ij}/\max TF_{ik})+1)/2$

where max is over all terms in *i*-th row of matrix **D**.

Disadvantage: term with high TF is in many D_i $\Rightarrow$ low P

Vector space model

IDF *inverse document frequency*

IDF for term t_j is defined as

$$IDF_j = \log(m/DF_j) + 1$$

where m is the number of documents in **D** and DF_j *(document frequency)* is a frequency t_j in **D**, i.e. the number of documents containing term t_j.

IDF is decreasing with the increasing number of documents containing the term.

Remark:

- for document ranking the base of the log is immaterial.
- IDF is really inverse w.r.t. DF.

Vector space model

- Behavior:

term occurs in all documents $\Rightarrow$ log(1) = 0 (term is one of the stop words)

term occurs only in 1 document $\Rightarrow$

$IDF = \log m + 1$

Ex.: $IDF = 2$ for $m = 10$ je, $IDF = 5$ for $m = 10\,000$, etc.

Vector space model

- A typical weighting is tf-idf weighting:

  $w_{ij} = TF_{ij} * IDF_j$ or $TD_{ij} = NTF_{ij} * IDF_j$

Notation in literature: tf-idf, tf.idf, tf x idf

Remark: it is not worthwhile to maintain too small w_{ij} (approaching the threshold).

- The best weights in Q:

  $q_k = (0{,}5 + (0{,}5 * TF_k)/\max TF) * IDF_k$

  where TF_k is term frequency of t_k in Q, max TF is maximum frequency of a term in Q and IDF_k is *IDF of* term t_k in **D**.
- Experimentally, tf-idf has been found to work well.

Vector space model

Special cases for *Q* and **D**:

- only a set of terms is given ⇒ $q_k = IDF_k$
- approximation of long queries ⇒ $q_k = TF_k$
- short documents ⇒ approximation weights by 0, 1
- long documents ⇒ retrieval unit is *passage*

Vector space model: problems

- Assumption: independency of terms (synonymy still not solved)
- Missing syntactic information (phrase structure, word order, proximity information)
- Missing semantics (e.g. word sense)
- History: part of the SMART system (1970)

Today: Apache Lucene – combines vector space and Boolean model

Vector space model in Boolean system - example of implementation

Assumptions:

- index file with inverted lists
- in inverted lists TF_{ji} (we model w_{ji})
- a file containing IDF_j
- file SCORE[1:m]
- term weights of query terms are equal to 1

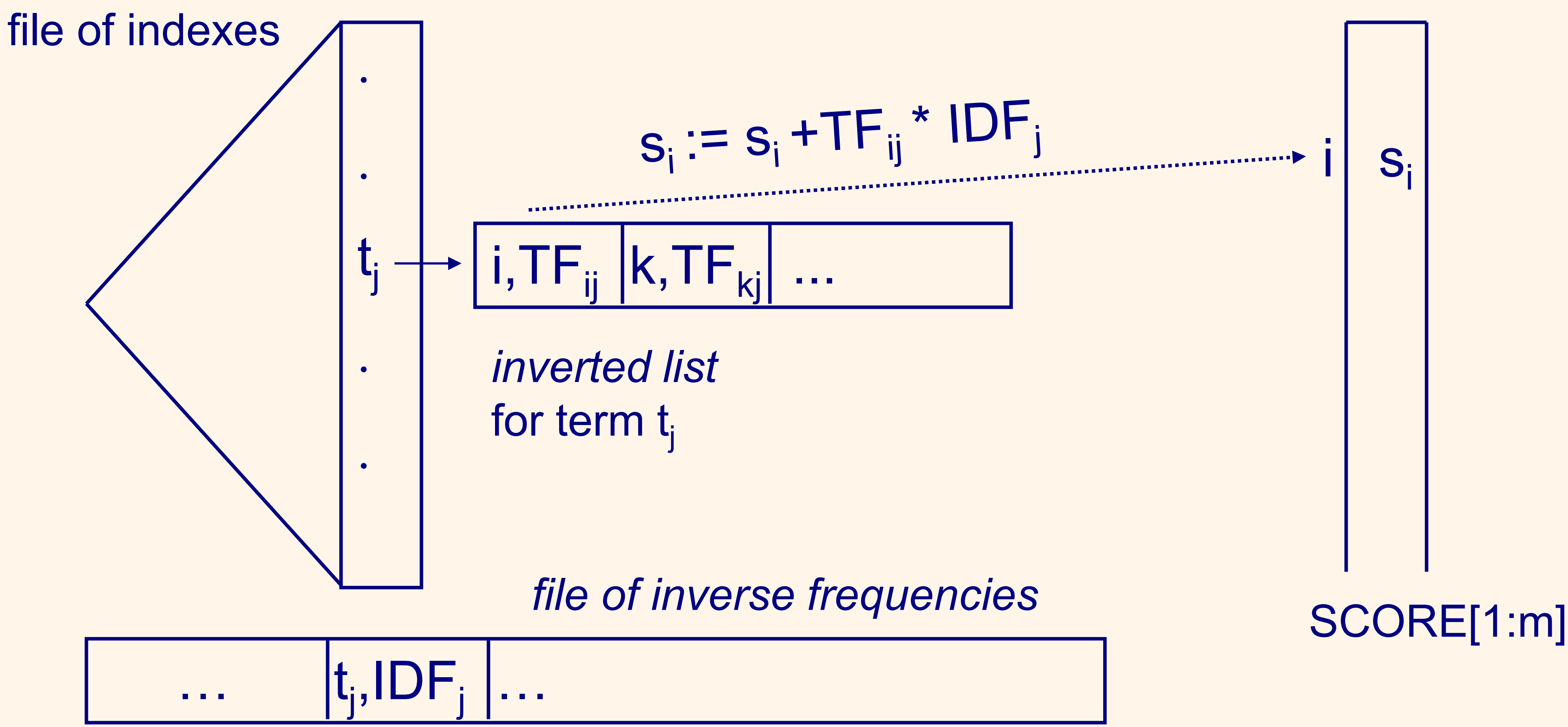
Algorithm:

(1) podle query terms přistupuj inverted lists.

(1.1) Oprav sums in SCORE

(2) Sort SCORE and return, e.g., 20 nejvyšších.

Vector space model in Boolean system - example implementation



Vector space model and signatures - example implementation

Assumptions:

- D_j has b_j blocks, query has Q terms
- signature file - there is a signature for each block
- file containing IDF_i (we model q_i in this way (*DF is* enough)
- file SCORE[1:20] (maintains the 20 higest)

Algorithm: For all D do:

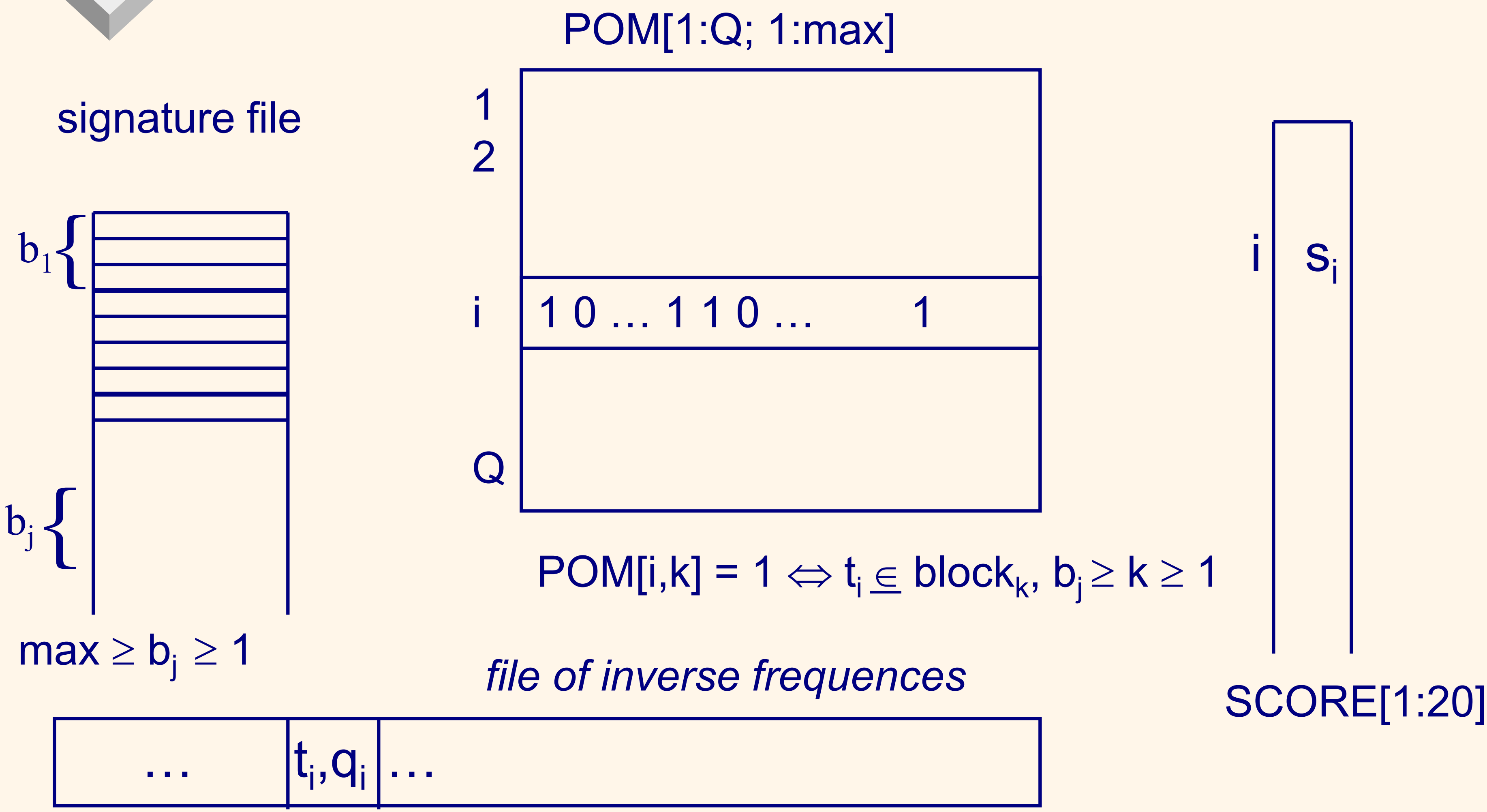
(1) Vynuluj POM.

(2) signature of each from b text blocks D compare with Q query signatures. Results store into POM.

(3) For each t_i query calculate $bc_i = \Sigma_{j=1\ldots bmax} POM[i,j]$

(4) Calculate $s = \Sigma_{i=1\ldots Q}(bc_i * q_i)/b$

Vector space model and signatures - example implementation



Indexing complexity by vector space model

- Vectors construction and indexing document with *n* units is O(*n*).
- indexing *m* such documents is O(*m n*).
- calculation of IDFs can be done in the same pass
- calculation of vector lengths is also O(*m n*).
- $\Rightarrow$ total time complexity is O(*m n*)

Example 1 – Text extender

```
SELECT journal, date, title
FROM ARTICLES
WHERE CONTAINS(article_text, '("database" AND
            ("SQL" | "SQL92") AND NOT "dBASE")') = 1;
```

Other functions: NO_OF_MATCHES (how often the search criteria are found in each text documen), RANK (rank value in answer based on a measure).

```
SELECT journal, title
FROM ARTICLES
WHERE NO_OF_MATCHES (article_text, 'database') > 10;
```

```
SELECT journal, date, titul, RANK(article_text, '("database" AND
("SQL" | "SQL92") )') AS relevant
FROM ARTICLES
ORDER BY relevant DESC;
```

possibility of different implementations

Example 2 – Fulltext in MySQL 5.1

Types of FT retrieval:

- Boolean
- FT with index

```
CREATE TABLE ARTICLES (
journal TEXT
article_text VARCHAR(200)
FULLTEXT (journal, article_text)
) engine=MyISAM

SELECT *
FROM ARTICLES
WHERE MATCH(journal, article_text)
AGAINST('database' IN NATURAL LANGUAGE MODE);
```

FULLTEXT is an index type

storage machine
other: InnoDB,…

Result sorting: implicitly by relevance

Example 2 – Fulltext in MySQL 5.1

Types of FT retrieval:

- Boolean
- FT with index

```
SELECT *
FROM ARTICLES
WHERE MATCH(journal, article_text)
AGAINST('+database –relational' IN BOOLEAN MODE);
```

Result sorting :

- + (AND), - (NOT), no operator (OR)
- implicitly no sorting

Techniques for “intelligent” IR

1. relevance feedback
   - direct feedback
   - pseudo feedback
2. query expansion
   - by „natural“ thesaurus
   - „artificial“ thesaurus

Advantages: increase R, only rarely P.

Relevance feedback

Intuition:

- vectors of relevant document and query are similar
- vectors of non-relevant document and query are not similar;

$\Rightarrow$ *query reformulation* based on the answer to query

- Assumptions: query vector $\vec{q}$

answer contains: relevant $D_1^r, \ldots, D_{mr}^r$

non-relevant $D_1^n, \ldots, D_{mn}^n$

Relevance feedback

$$\vec{q}\,' = \alpha\vec{q} + \frac{\beta}{m_r}\Sigma_{i=1\ldots mr}\vec{D}_i^{\,r} - \frac{\gamma}{m_n}\Sigma_{i=1\ldots mn}\vec{D}_i^{\,n}$$

for α=1 Rocchio 71

$$\vec{q}\,' = \alpha\,\vec{q} + \beta\,\Sigma_{i=1\ldots mr}\vec{D}_i^{\,r} - \gamma\,\Sigma_{i=1\ldots mn}\vec{D}_i^{\,n}$$

for $\alpha = \beta = \gamma = 1$ Ide 71

$$\vec{q}\,' = \alpha\,\vec{q} + \beta\,\Sigma_{i=1\ldots mr}\vec{D}_i^{\,r} - \gamma\,\vec{D}_1^{\,n}$$

where α, β, γ are appropriate constants

Relevance feedback - incrementally

REPEAT

1. System retrieves D with maximal SIM(Q,D);
2. User marks D as relevant or non-relevant;
3. IF D is relevant THEN D goes to the output list;
4. $\vec{q}$ is modified by $\vec{D}$;

UNTIL φ

Query modification:

$$\vec{q}_{j+1} = \begin{cases} \alpha \vec{q}_j + \beta \vec{D}_j & D_j \text{ is relevant} \\ \alpha \vec{q}_j - \gamma \vec{D}_j & D_j \text{ is non-relevant} \end{cases}$$

Remark: a D that has not yet been selected is always selected.

Relevance feedback – other possibilities

reweighting terms: increasing term weights in relevant documents and decreasing term weights in non-relevant documents

pseudo-feedback: consider the first *k* documents as relevant and then do relevance feedback (query reformulation).

Extending query by thesaurus

- *Thesaurus* (in Latin treasure, treasury) provides synonym information and about semantically related words and phrase.
- Ex.: Eurovoc – for law and legislation, is from 2005 also for Czech.

Thesaurus

Expressions using thesaurus (standard ISO-2788)

NT('text')	NARROWER TERM	one level narrower term
NT('text',n)		*n* levels narrower terms
NT('text',*)		all narrower terms
BT('text')	BROADER TERM	one level broader term
BT('text',n)		*n* levels broader term
BT('text',*)		all broader terms
TT('text')	TOP TERM	
SYN('text')	SYNONYMS	
PT('text')	PREFERRED TERM	
RT('text')	RELATED TERMS	

Thesaurus

Other relationships:

USE – to a given term assigns its preferred term,

UF (USE FOR) – to a given term assigns its synonymous (non-preferred) term

SN (scope note) - note attached to the given term

Other standard (for text collections):

ANSI Z39.58 Common Command Language for Online Interactive Information Retrieval – developer by institution NISO (National Information Standards Organization).

Remark: real languages are only similar to these standards

Example: Wordnet

- Lexical database of semantic relationships between words (of English, …, Czech).
- developed by Prof. George Miller and his team at Princeton university.
- 150,000 English words.
- Nouns, verbs, adjectives, and adverbs are grouped into cca 110,000 set of synonyms called *synsets*.

Example: Wordnet

Examples of conceptual relations among synsets:

- antonyms (have oposite meanings): wet $\rightarrow$ dry, young $\rightarrow$ old
- semantically similar to: dry $\rightarrow$ parched
- reason: killing $\rightarrow$ death
- holonymy : chapter $\rightarrow$ text (be part of)
- meronymy: computer $\rightarrow$ cpu (has as a part)
- hyponymy (subordinate notions): tree $\rightarrow$ plant (specialization)
- hyperonymy (superordinate notions): fruit $\rightarrow$ apple (generalization)

Example: Wordnet

- Measuring semantic similarity and correlations introduced for WordNet by Pederson, et al in r. 2005 – (software WordNet::Similarity)
- similarity coefficients
  - based on paths lengths:

    Lch, wup, Path
  - based on information content:

    res, lin, jcn
- relatedness measures
  - hso, lesk, vector

Conclusion

Current (new) applications:

- text classification
- text extraction (summarization)
- digital libraries
- Web retrieval
- multilingual environment
- spam detection
- text plagiarism detection