Query languages 2 (NDBI006)

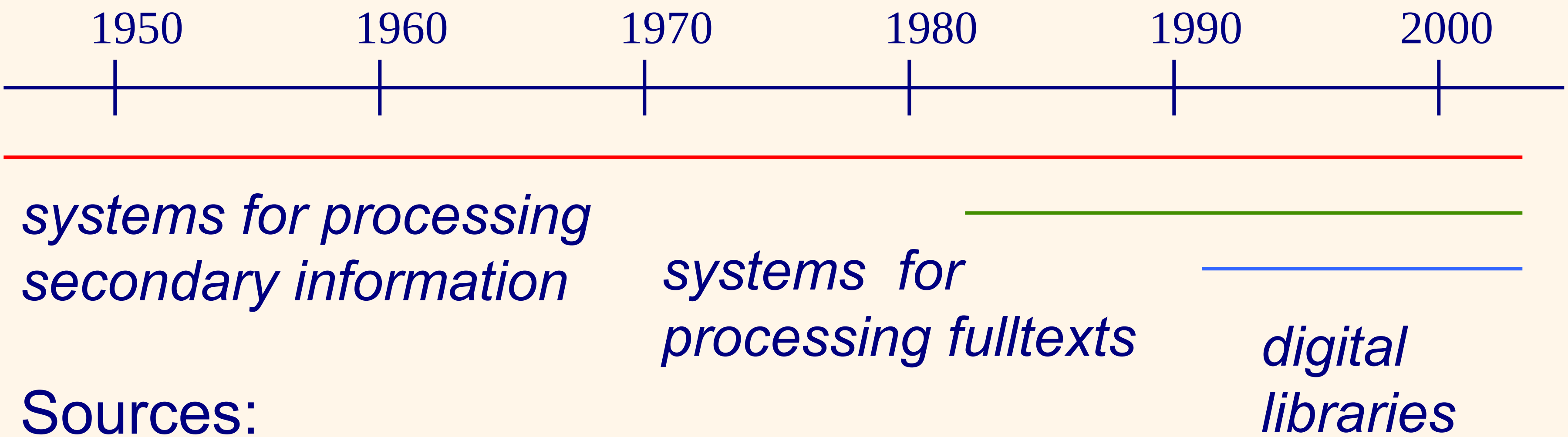
Information Retrieval

Jaroslav Pokorný

MFF UK, Praha

pokorny@ksi.mff.cuni.cz

Development of IS



Sources:

- formation of texts directly in a computer
  - Need: searching, not only browsing,
  - not always possible to index documents manually
- development of big memories (CD ROM, WORM)
- development of communications (Internet)

Content

Information retrieval

database - a collection of documents (unstructured, no schema)

query - requirement formulated in a language is usually entered with a text sample (word, expression, part of a word, or even the entire text) or several samples (*conjunctive query*)

More generally: Boolean expressions

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

hit relevance – extent measure, how the hit matches the user request

Answer restriction - maximum M
- at most M most relevant ones
- entering a threshold value Θ

Information retrieval

Field: **Information Retrieval** (IR)

IR is all about finding what you want when what you want is hidden in the mass of what you don't want.

More precisely:

To find to the query relevant documents

Field: **Information Filtering**

To retrieve to the document D profiles in such way, that D is for them relevant.

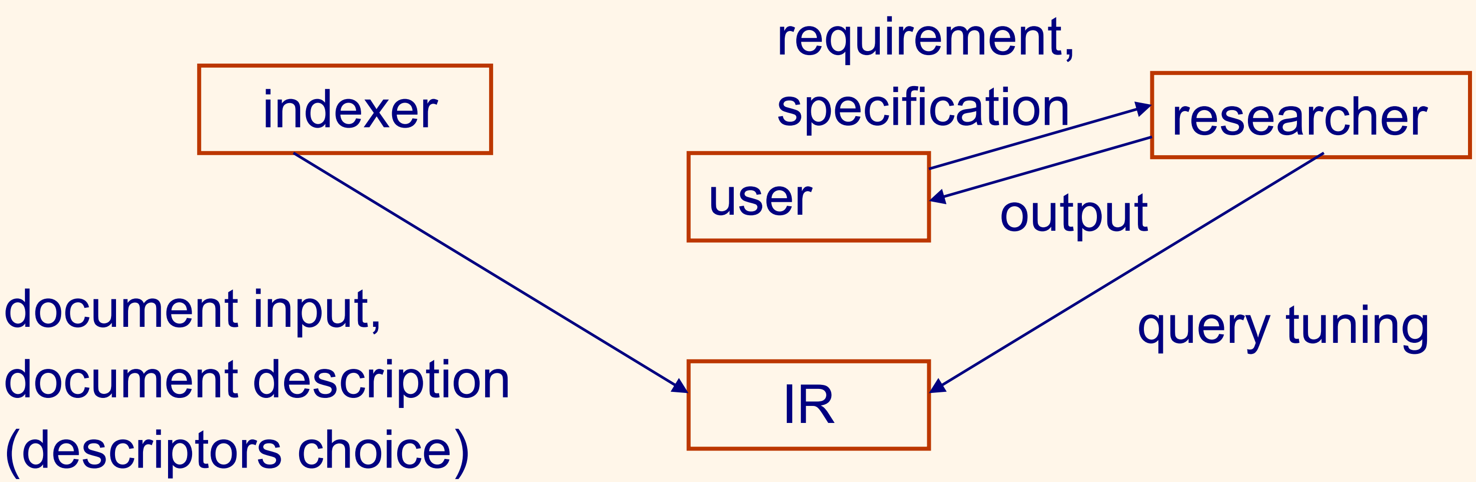
IR - basic architecture

Subsystems: making text accessible (1)

text delivery (2)

(1) see information services

secondary information vs. fulltexts



historical model

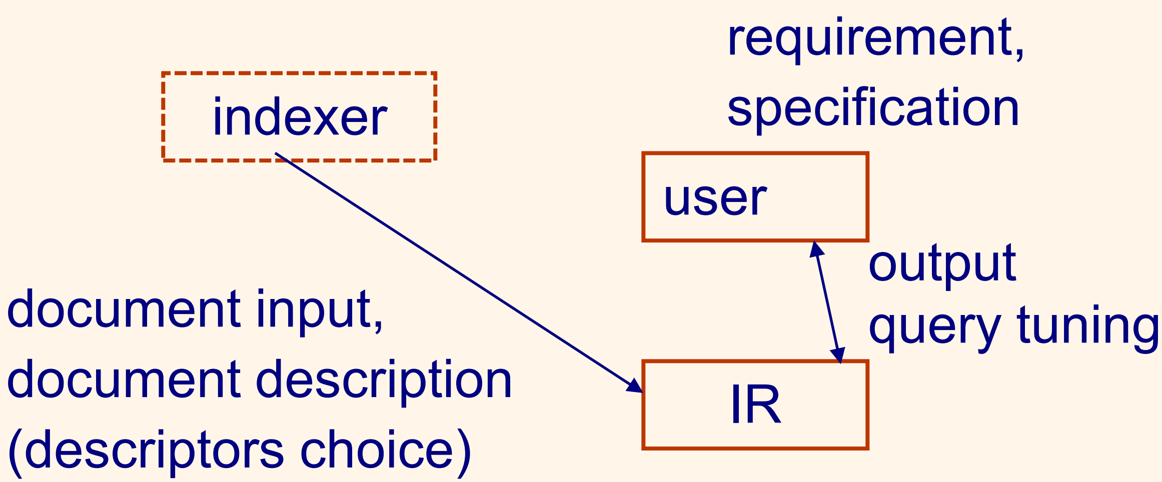
IR - basic architecture

Subsystems: making text accessible (1)

text delivery (2)

(1) see information services

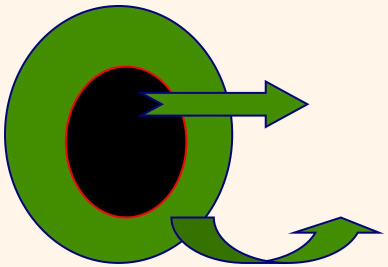
secondary information vs. fulltexts



current model

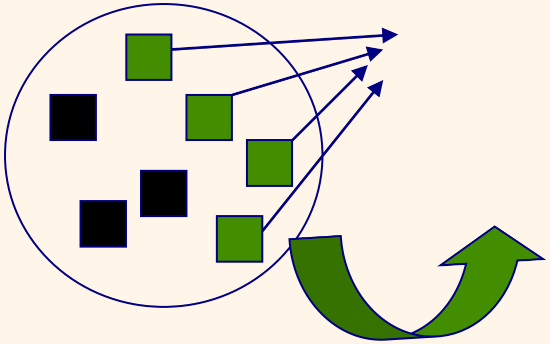
Measuring relevance

recall R



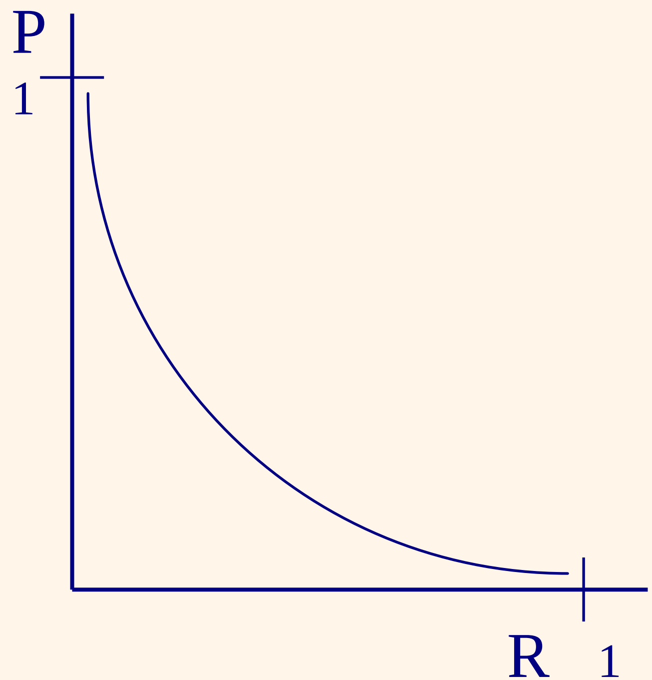
$$R = \frac{\text{\#retrieved relevant documents}}{\text{\#relevant documents in the set of all documents}}$$

precision P



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

Measuring relevance



precision-recall curve

Boolean model

- Document representation: as a set of terms
- Querying:
  - formally: with Boolean expressions
  - style: exact matching
- Finding terms - practice:
  - removal of stop-words (very common words such as “a”, “an”, “the”, “it” etc.) from the set of terms

    results in reduction 30-50% (C.J. van Rijsbergen)
  - linguistic processing (tokenization)
- Creation of the inverted index

Boolean model

One of possible syntaxes:

<term>

<attribute_name> = <attribute_value> /comparison/

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

X AND Y	retrieve D, containing X and Y as well.
X OR Y	retrieve D, containing either X or Y.
X XOR Y	retrieve D, containing either X or Y but X AND Y is not TRUE
NOT Y	retrieve D, not containing Y
X adj Y	retrieve D, in which X occurs followed by Y
X (n)words Y	retrieve D, in which X occurs followed by Y in maximal distance *n* words
X sentence Y	retrieve D, in which X and Y occur in the same sentence

Boolean model

Language

. for any character.

* character followed by * corresponds to any number of occurrences (including zero) of this character. For example, xy* corresponds to x, xy, xyy etc.

+ character followed by + corresponds to any number of occurrences (except of empty) of this character. For example, xy+ corresponds to xy, xyy, xyyy etc.

[] Characters in [] correspond to any single character, který is in parentheses given, but not to another. For example, [xyz] corresponds to x, y or z.

[^] ^ at the beginning of a string in [] means negation (not). For example, [^xyz] corresponds to any character except x, y or z.

[-] – between characters in [] indicates range characters. For example [a-x] corresponds to any character between a and x.

Boolean model: P vs. R

- By refining the query in Boolean model, we obtain greater P, but smaller R.

Example: experiment (Blair, Maron,1985) – 40 000 legal texts

Goal: not only high P, but also R.

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

Problem of synonyms – the use of natural language, cannot be captured by a thesaurus.

Example: accident, mishap, collision, car accident, "something happened there", ...

- automatic indexing does not eliminate these problems

Boolean model: problems

- Thus far, our queries have all been Boolean.
  - Documents either match or don’t.
- Good for expert users with precise understanding of their needs and the collection.
  - Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users.
  - Most users incapable of writing Boolean queries (or they are, but they think it’s too much work).
  - Most users don’t want to wade through 1000s of results. This is particularly true of web search

Boolean model: problems

What affects the P and R relationship?

Problems with manually indexed systems:

uncertainty

- in indexing — influence of the indexer
- in the choice of terms for query — influence of the user

Example: p_1, p_2 probabilities, that user uses terms t_1, t_2

q_1, q_2 probabilities, that terms t_1, t_2 occur in D

$\Rightarrow$ p, that the user chooses t_1, t_2 and D with t_1, t_2 is selected, is

$$p_1 * p_2 * q_1 * q_2$$

For example, R = 0,6 * 0,7 * 0,5 * 0,6 = 0,126 $\Rightarrow$ R < 13%

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

$\Rightarrow$ from 1000 relevant Ds, only 1 is chosen!

Boolean model: problems

prediction criterion - how to ensure agreement between the selection of terms for query and documents (today: similarity of ontologies)

- method: removing uncertainty

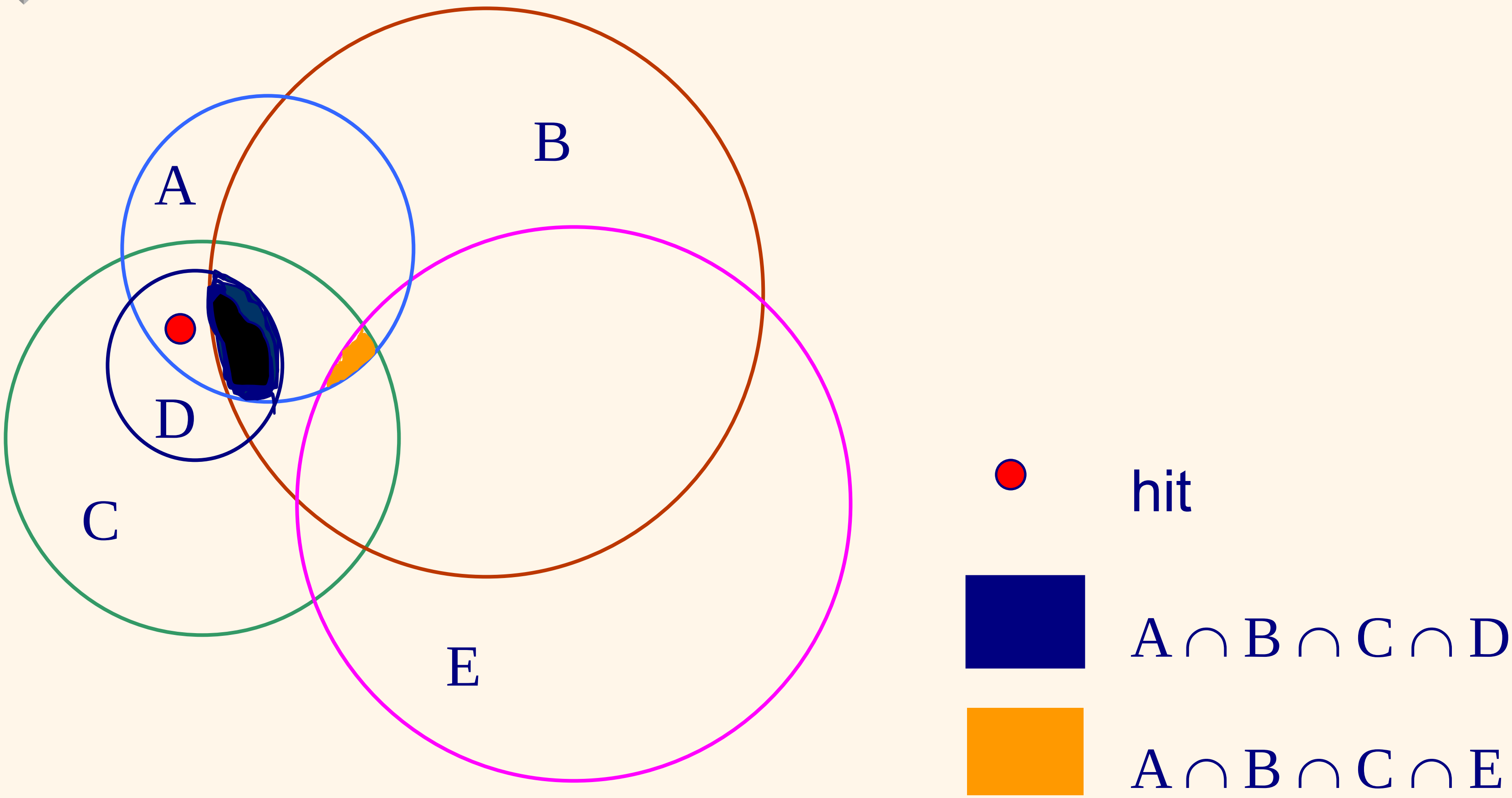
maximum criterion - 20-50 hits can be handled

Problems: AND gives too few; OR gives too many

Problems with fulltext DB :

- DB size (vs. maximum criterion)
- selecting terms for query
- revaluation of the elimination of indexers
- the indeterminacy of the questioner remains
- unilateral behavior of the user
- tendency to change the last decision, keep first steps

Boolean model: problems



Boolean model: problems

Solving uncertainty in the choice of terms for query:

- we find D with high relevance for user (D is known + is known, that occurs in DB),
- terms for query are selected from D,
- removing terms or their replacement by disjunctions.

$\Rightarrow$ reducing the indeterminacy of the user.

Boolean model: problems

Solution of unilateral behavior of the user by weighting:

Example:	*terms*		*probability (weight)*
	Author: Pokorný		0,3
	Date: 1995-1999		0,7
	Journals:	CW	0,2
		Artificial Intelligence	0,5
		ERCIM News	0,2
	Keywords:	XML	0,6
		databases	0,8
		query languages 1	0,9

Total number of conjunctive queries is 255.

Boolean model: problems

Products of probabilities for

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

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

Boolean model: other problems

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

    D not containing only one z the terms listed will not be selected.
  - A OR B OR C OR D OR E

    Ds containing only one from the terms listed are understood as equally significant as documents containing all terms listed.
- It does not allow control of the output size.
- all Ds satisfying the query are seen as equally important; it is not possible to rank them by degree of relevance.

Boolean model: other problems

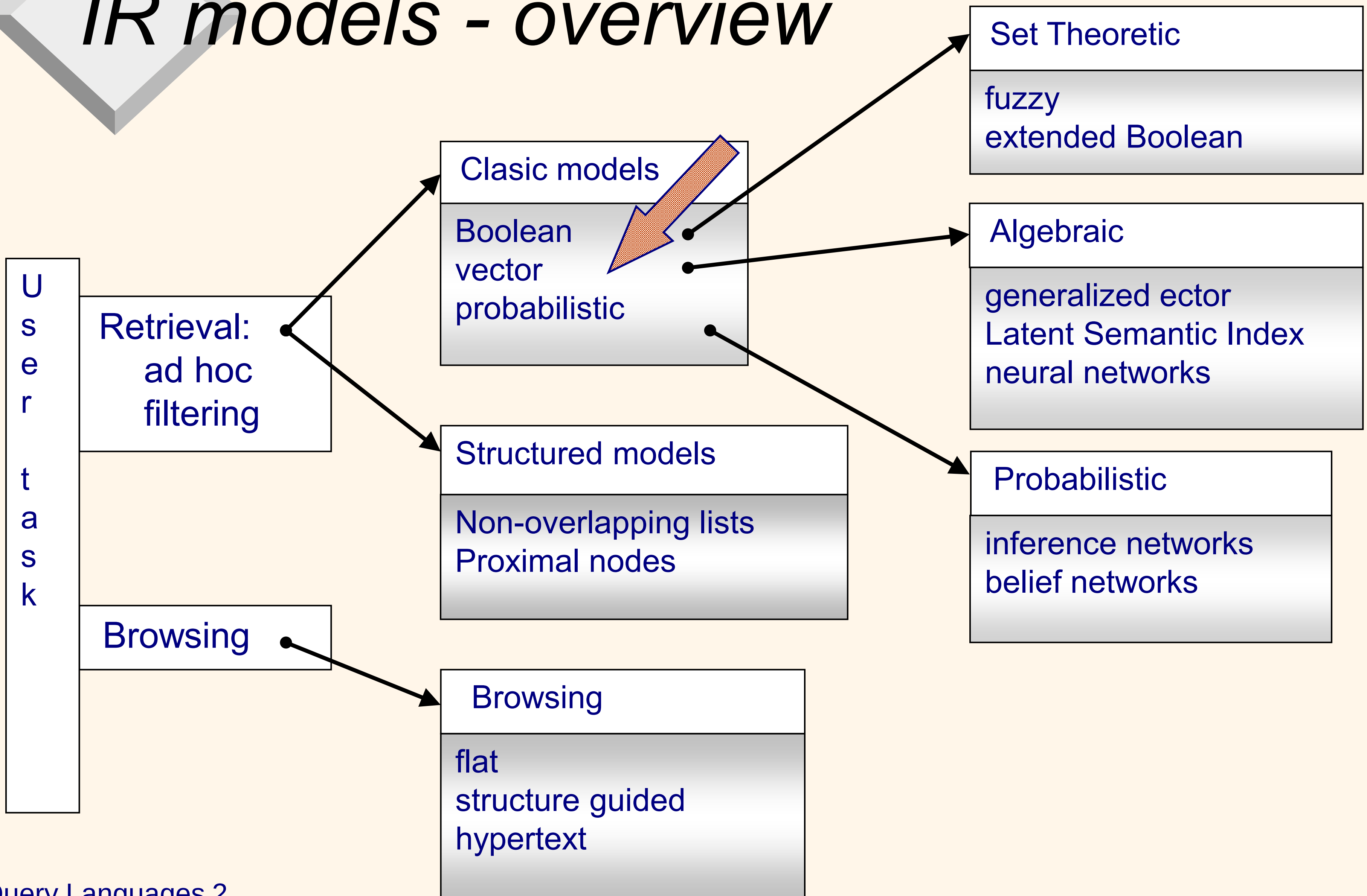
- It is difficult to implement automatic feedback, i.e. automatically modify query based on D marked in answer as relevant.
- Expressive power of the Boolean model is restricted. Any set {D} describable by terms, can be, in principle, selected by an appropriate Boolean query. But it is not guaranteed, that for any set of documents {D} that are of interest to the user, it is simple to formulate a Boolean query in practice.
- More of an art than a science.

What next?

Thesis:

Classical Boolean systems can be extended by functions affecting the maximum criterion; however, it is not possible to simultaneously reach high P and R as well without additional information.

IR models - overview



Vector model

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

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

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

w_{ij} is a weight assigned to term t_j in identification of document D_i.

D is representable by term-document matrix

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

Zero means the term has no significance in the D_i or it simply doesn't exist there.

Vector model

- querying:
  - formally: with a query vector
  - partial match search

    method: by 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 measure the degree of 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 model

Angle vs. distance

- Why not distance?
- Experiment: we take document D and connect it once more to D. Document D′ is created.

  "Semantically" D and D′ have the same content.
- The Euclidean distance between points in space between D and D′ (point spaces) would be large.
- Angle between D and D′ (as vectors) is 0, i.e., it corresponds to maximal similarity.
- Key idea: rank documents D in decreasing order of the angle between query and document.
- Appropriate measure: cosine – descending function for the interval [0^{o}, 180]. Then use cosine(query, D).

Vector model

Similarity coefficient (angl. similarity) of query Q and document D_i

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

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

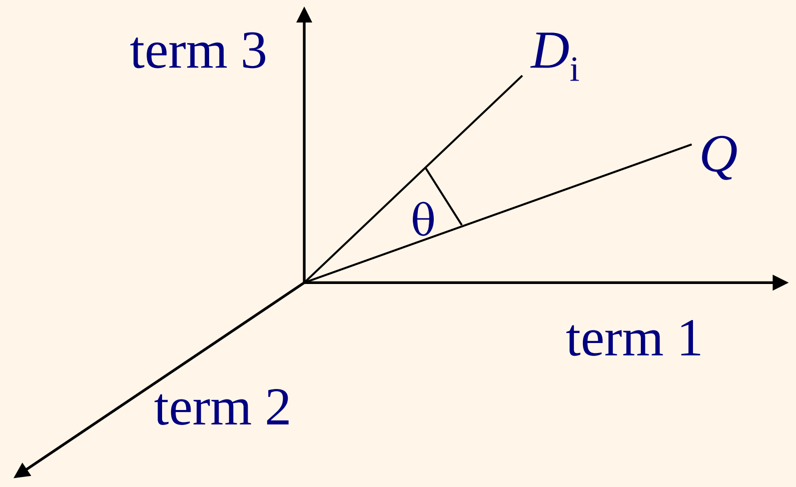
The divisor in (b) is the normalization factor,

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

Postulate: documents that are in the vector space "close to each other" tell about the same things

Vector model

geometric interpretation



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

For all three cases *Sim* =

- $|Q \cap D_i|$
- $(|Q \cap D_i|)(\sqrt{|Q|} * \sqrt{|D_i|})$
- $2(|Q \cap D_i|)(|Q| + |D_i|)$

Vector model

Advantages: R and P can be increased by up to 20%.

Pragmatic approach: one-word terms + appropriate weighting method

TFi_j term frequency t_j in document D_i

NTF_{ij} normalized term frequency t_j in document D_i

$((TF_{ij}/\max TF_{ik})+1)/2$

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

Disadvantage: a term with high TF in many $D_i \Rightarrow$ smaller P

Vector model

IDF inverse document frequency of term decreases with the increasing number of documents to which the term is assigned.

IDF for term t_j is defined as

$$\mathrm{IDF}_j = \log(m/\mathrm{DF}_j) + 1$$

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

Remark:

- for document ranking the logarithm base is not important
- IDF is really inverse w.r.t. DF.

Vector model

Behavior:

- term occurs in all documents $\Rightarrow$ log(1) = 0 (term belongs to words with no significance)
- term occurs only in 1 document $\Rightarrow$ IDF = log m +1

Example: IDF = 2 for m = 10 , IDF = 5 for m = 10 000, etc.

Intuition: importance of a term is high when it occurs a lot in a given document and rarely in others. In short, commonality within a document measured by TF is balanced by rarity between documents measured by IDF.

Vector model

$\Rightarrow$ TF-IDF matrix

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

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

Remark: it is not good to keep too small term weights (to the threshold value).

- Q can be entered as a document.
- The best weights for 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**.

Vector model

Special cases for Q and **D**:

- only set of terms is specified $\Rightarrow$ q_k = IDF_k
- approximations of long queries: q_k = TF_k
- short documents $\Rightarrow$ approximation of weights with 0, 1
- long documents $\Rightarrow$ a unit of selection is a *passage*

Vector model: problems

- assumption: term independency
- missing syntactic information (phrases, word oder, distances)
- missing semantics: polysemy, synonymy are still not solved

History: a part of the SMART system (1970)

Today:

- Apache Lucene – combining vector and Boolean model
- OpenSearch (software) (2021) – based on Apache License 2.0

Vector model in a Boolean system - example of implementation

Assumptions:

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

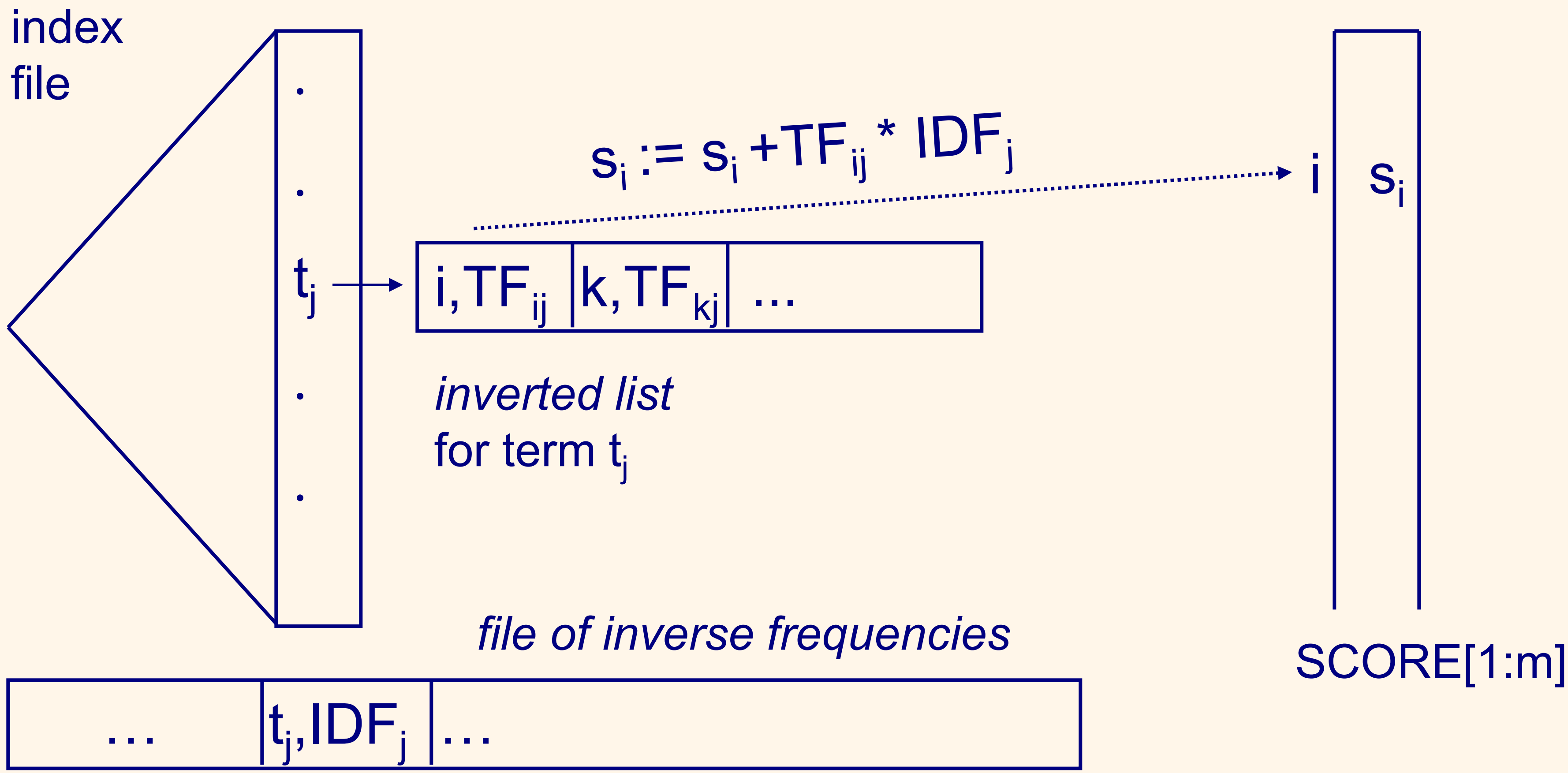
Algorithm:

(1) According to query terms access inverted lists.

(1.1) Change sums in SCORE.

(2) Order SCORE and return, e.g., 20 highest.

Vector model in Boolean system - example of implementation



Vector model and signatures – example of implementation

Assumptions:

- D_j has b_j blocks, the query has Q terms
- signature file - for each block there is a signature
- file containing IDF_i (we use them to model q_i - DF is enough)
- file SCORE[1:20] (the top 20 are maintained)

Algorithm: Do for all D:

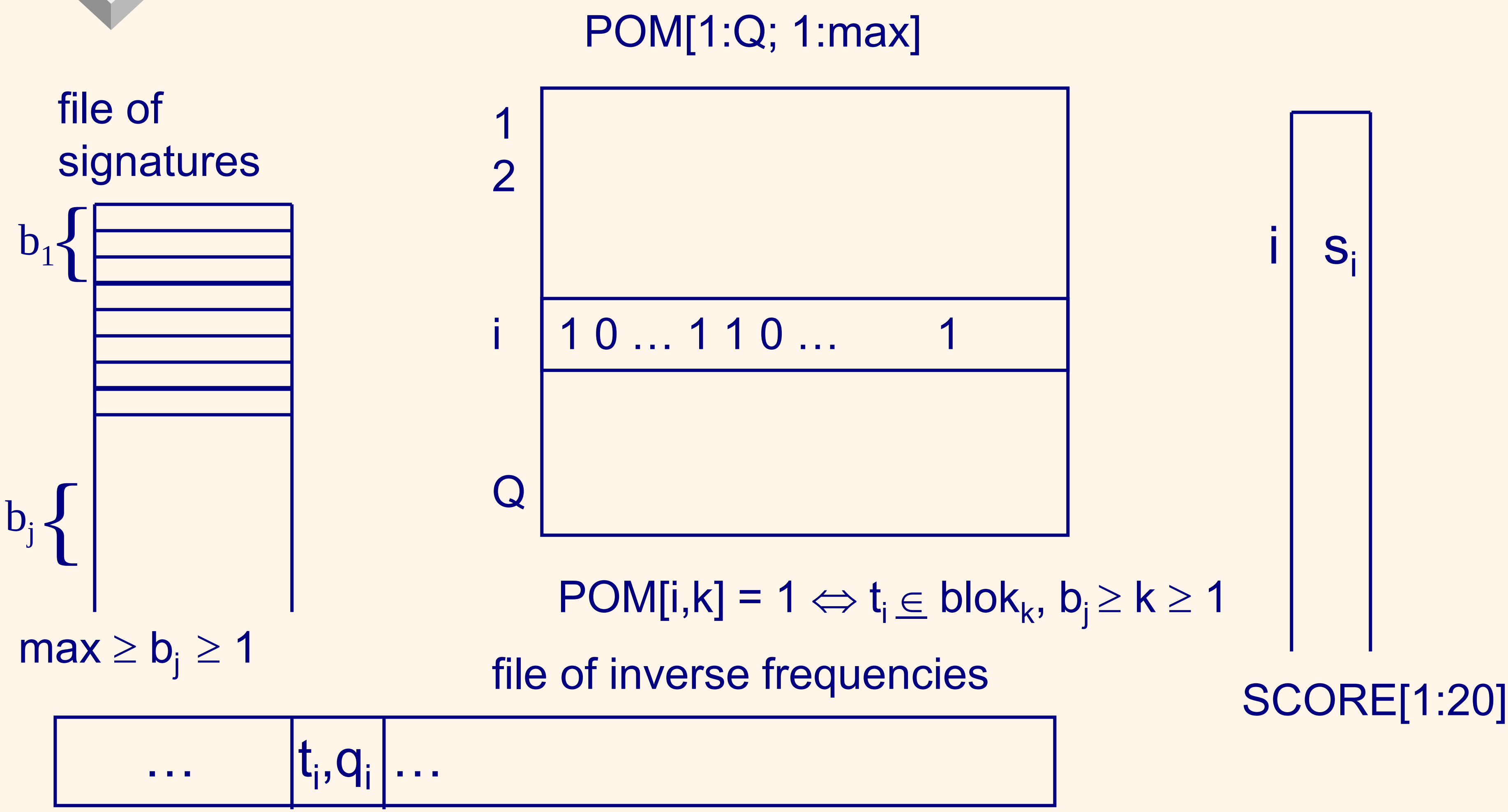
(1) Reset POM.

(2) Signature of each from *b* blocks of text D compare with Q signatures of the query. Save results to POM.

(3) for each t_i of the 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 model and signatures – example of implementation



Complexity of indexing by vector model

- creating vectors and indexing document with *n* units is O(*n*),
- indexing *m* such documents is O(*m n*),
- counting IDFs can be done in the same pass,
- computing the lengths of vectors is also O(m n).
- $\Rightarrow$ total time complexity is O(*m n*).

Example 1 – Text extender in DB/2

```
CREATE TABLE ARTICLES(
    journal          VARCHAR(50),
    title            VARCHAR(50),
    date             DATE,
    article_text     FULLTEXT)

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

Example 1 – Text extender in DB/2

Other functions: NO_OF_MATCHES (number of times the specified pattern occurred in the text), RANK (based on some measure).

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

```
SELECT journal, date, title, 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 fulltext (FT) searching:

- Boolean
- FT with index

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

FULLTEXT is an index type

Storage machine
other: InnoDB,…

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

Sorting results: implicitly by relevance

Example 2 – Fulltext in MySQL 5.1

Types of FT searching:

- Boolean
- FT with index

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

Sorting results:

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

Technics for “intelligent” IR

1. feedback
   - direct feedback
   - pseudo-feedback
2. extending query
   - „natural“ thesaurus
   - „artificial“ thesaurus

Advantages: increase R but rarely P.

Feedback

Intuition:

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

$\Rightarrow$ query reformulation based on the query answer

Assumptions: query vector $\vec{q}$

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

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

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

Feedback - incrementally

REPEAT

1. System selects D with max. SIM(Q,D);
2. The 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:

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

Feedback – other possibilities

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

pseudofeedback: assume the first *k* documents as relevant and modify the query according to them.

Query extension with thesaurus

- thesaurus (lat. treasure, treasure) provides information about synonyms and semantically related words and phrases.
- Example: Eurovoc – for area of law and legislation, from 2005 there is also for Czech.

Thesaurus

Expressions using the thesaurus (standard ISO-2788)

NT('text')	NARROWER TERM o level narrower term
NT('text',n)	narrower terms o *n* levels
NT('text',*)	all narrower terms
BT('text')	BROADER TERM o level broader term
BT('text',n)	broader terms o *n* levels
BT('text',*)	all broader terms
TT('text')	TOP TERM – the broadest term
SYN('text')	SYNONYMS - synonyms
PT('text')	PREFERRED TERM preferred term
RT('text')	RELATED TERMS - related terms

Thesaurus

Other relations:

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

USE - to the given term assigns its preferred term,

UF - to the given term assigns its synonymous (non-preferred) term

Other standard (for text DB):

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

Remark: real languages are only similar to these standards

Example: Wordnet

- more detailed database of semantic relationships between words (for English, …, Czech).
- developed by Prof. George Miller and his team at university in Princeton.
- about 150,000 English words.
- Nouns, adjectives, verbs and adverbs arranged into cca 110,000 synonymous sets called synsets.

Example: Wordnet

Examples of relationship types:

- antonyms (opposites): in front$\rightarrow$ behind
- atributation: charity $\rightarrow$ good (from noun to adjective)
- similarity: unconditional $\rightarrow$ absolute
- cause: killnig $\rightarrow$ death
- holonyms: chapter $\rightarrow$ text (to be a part)
- meronyms: computer$\rightarrow$ cpu (to be a part)
- hyponyms (subordinate terms): tree $\rightarrow$ plant (specialization)
- hyperonyms (superordinate terms): fruit$\rightarrow$ apple (generalization)

Example: Wordnet

- Measuring semantic similarity and relatedness introduced for WordNet by Pedersen, et al in 2005 – (software WordNet::Similarity)
- Similarity coefficients
  - Based on path lengths:
    Lch, wup, Path
  - Based on information content:
    res, lin, jcn
- relatedness coefficients:
  - hso, lesk, vector

Conclusion

Current (new) applications:

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