course:

Searching the Web (NDBI038)

Searching the Web and Multimedia Databases (BI-VWM)

lecture 2:

Boolean model of information retrieval

prof. RNDr. Tomáš Skopal, Ph.D.
Department of Software Engineering, Faculty of Mathematics and Physics, Charles University, Prague
Department of Software Engineering, Faculty of Information Technology, Czech Technical University in Prague

Today's lecture outline

- information retrieval
  - the problem, preliminaries
  - text preprocessing
  - quality of retrieval (the effectiveness)
- classic Boolean model
  - data model
  - queries
  - implementation (indexing)
- extended Boolean model
  - term weighting

Information retrieval – the problem

- for this lecture, consider the web as an extremely huge collection of **plain text documents** (the text in web pages)
- i.e., forget other than full-text content (HTML tags)
  - no hyperlinks to other web pages
  - no embedded multimedia, no CSS styles, no PHP
  - no web sites (group of related web pages) or other web page "clusters"
  - no structure of the text (paragraphs, sections, chapters)

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
<head>
<META http-equiv=content-type content=text/html;charset=UTF-8>
<meta name="description" content="description"/>
<meta name="keywords" content="keywords"/>
<meta name="author" content="author"/>
<link rel="stylesheet" type="text/css" href="http://siret.ms.mff.cuni.cz/default.css" media="screen"/>
<title>SRG</title>
</head>
<body>
<div class="container">
   <div class="header">
   <img src="http://siret.ms.mff.cuni.cz/img/siret_logo1.gif" align="left" style="position:absolute; padding-top:20px; left:5px;">
      <div class="title" style = "">
         <h1><b>SRG</b> - Siret Research Group</h1>
      </div>
      <div class="navigation">
         <a href="http://siret.ms.mff.cuni.cz/index.php">Home</a>
         <a href="http://siret.ms.mff.cuni.cz/pubs.php">Publications</a>
         <a href="http://siret.ms.mff.cuni.cz/projects.php">Supporting projects</a>
         <a href="http://siret.ms.mff.cuni.cz/members.php">Members</a>
         <a href="http://siret.ms.mff.cuni.cz/links.php">Links</a>
         <div class="clearer"><span></span></div>
      </div>
   </div>
   <div class="main">
   <div class="main">
      <div class="content">
         <h1>Siret</h1>
```

```
SRG - Siret Research Group

Siret (SImilarity RETrieval) research group (SRG) was founded in 2006 at the Department of Software Engineering, Charles
University in Prague.

Characterization
The SRG (SIRET research group) deals with database methods for efficient and effective similarity search in multimedia
databases. SRG engages three areas of interest - general indexing methods in metric and nonmetric spaces, biological
applications of the similarity search (RNA, proteomics, mass spectrometry …) and indexing image descriptors databases
involving complex similarity models. In the first area, SRG is developing an approximate similarity search framework,
allowing users to supply arbitrary similarity functio. This allows the users to model the similarity without restrictions
given by, e.g. , metric axioms. In the second area, SRG has proposed several models for representation of protein
tertiary structure and similarity functions for their effective and efficient classification and interpretation.
Currently, we are also working on similarity models of RNA structures. In the third area we have initiated the research
of indexability of image data while employing advanced similarity measures based on quadratic forms. Together with the
basic research, SRG is developing SIRET web engine, a complex and extendible software system for similarity search in
complex unstructured databases.
```

Information retrieval – the problem

- how to search collection of full texts?
  - one could use string matching algorithms, like Knuth-Morris-Pratt, Boyer-Moore, Aho-Corasick, etc.
  - sufficient? NO!
    - extremely slow (whole collection must be searched, no index)
    - limited in retrieval possibilities (simple pattern/substring) e.g., search for not-appearing text, or word close to other word
- solution?
  - retrieval model + indexing
    - **Boolean models + inverted index (today)**
    - Vector models + inverted index (next lecture)
    - Probabilistic models

Information retrieval – preliminaries

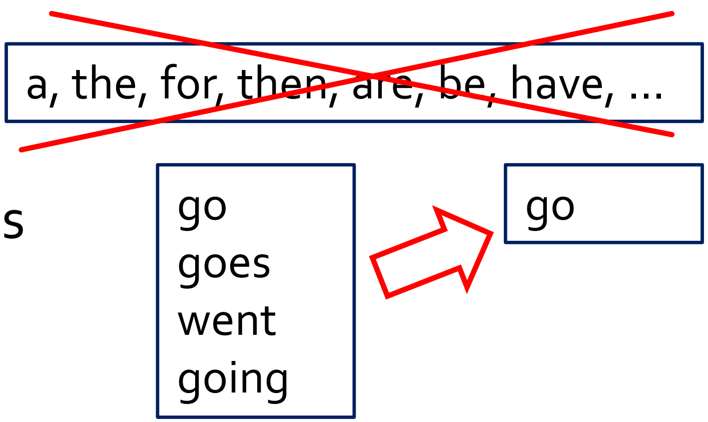
- basic definitions
  - **document** = entity containing full text
    - in a broader meaning – full-text annotation of another (multimedia) entity/document
  - **collection** = set of documents
  - **term** = word (phrase) appearing in a document
  - **vocabulary** = set of all distinct terms appearing in documents of a collection

- example
  - collection of 1000 newspaper articles
  - each article containing 1000 of words in total (50 **distinct** terms)
  - the vocabulary contains 10,000 **distinct** terms

Information retrieval – preliminaries

- preprocessing of the collection
  - without preprocessing, the vocabulary would contain too many terms, e.g., half a million (language-dependent)

1) need to remove some terms
   - stop words – very frequent ones, could be also numbers, names, etc.
2) need to simplify the remaining terms
   - stemming (lemmatization), other linguistic preprocessing



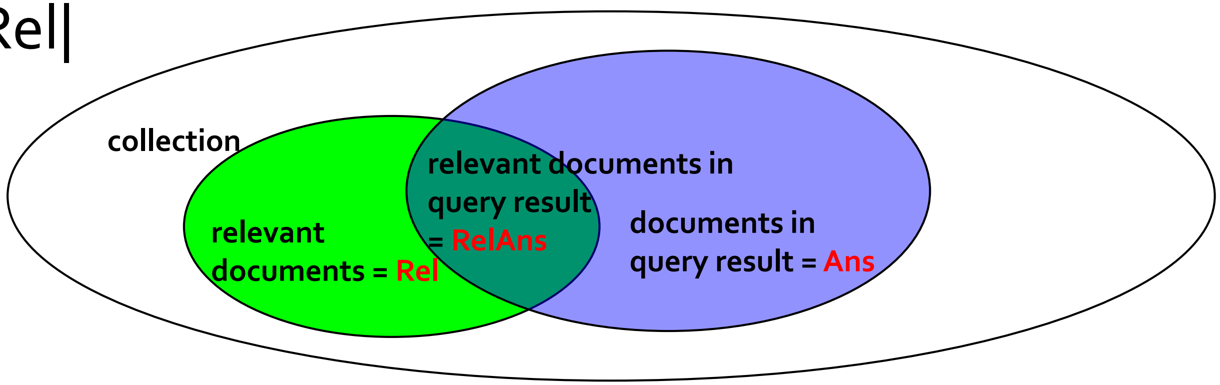
- we get smaller vocabulary, e.g., 20% of original terms
  - more compact storage + faster search
  - more effective search (match of stems is better than exact match)

Information retrieval – quality of retrieval (effectiveness)

- quality of a retrieval system
  - effectiveness = the measure of user's satisfaction with the result
    - based on determining relevant documents in the query result
- relevant document
  - given a particular query and collection, relevant documents should be within the query result of an examined system
  - relevancy of a document could be determined when
    - annotated collection is used
    - referential retrieval system is used
- extrapolation assumption
  - a system that is effective on an evaluated collection should be effective also on an unknown collection

Information retrieval – precision and recall

- two complementary measures
  - **precision** = probability that document in the result is relevant
  - **recall** = probability that a relevant document is in the result
- formally
  - P = |RelAns| / |Ans|
  - R = |RelAns| / |Rel|



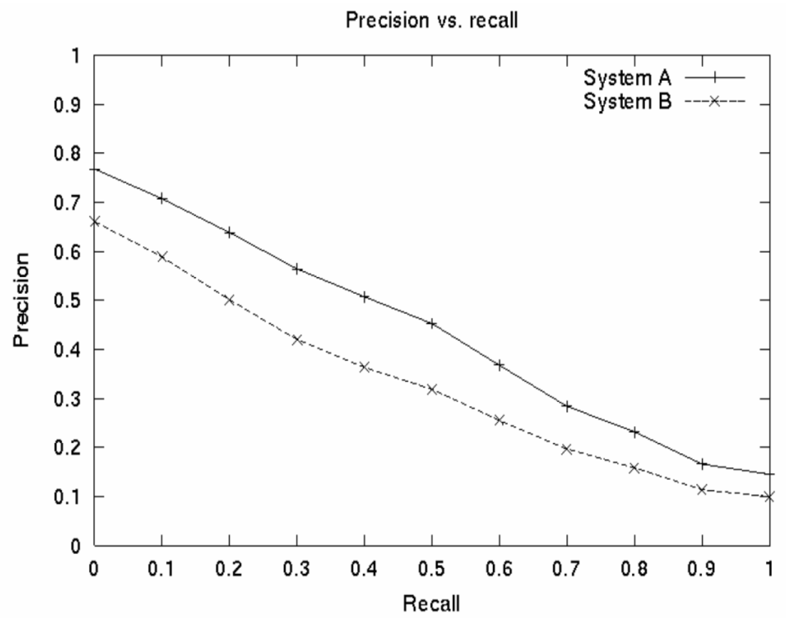
Information retrieval – relative precision/recall

- let QR_{ref} is the desired query result of a referential system
  - e.g., directly desired answer (annotated queries)
- let QR_{sys} is query result obtained by the evaluated system

- **false alarm** (false hit)
  - a document $d_j \in QR_{sys}$ and $d_j \notin QR_{ref}$
    - i.e., should not be retrieved but it was
  - the more false alarms, the lower **relative precision**
- **false dismissal** (false drop)
  - a document $d_j \notin QR_{sys}$ and $d_j \in QR_{ref}$
    - i.e., should be retrieved but it was not
  - the more false dismissals, the lower **relative recall**

Information retrieval – P-R curve

- the precision-recall curve shows the dependence of precision and recall
  - gives more complex characteristics of the system
  - enlarging query result is considered (not just single result)
- 11 standard levels of recall
  1) let's have a ranked query result
  2) enlarging the query result till a certain level of recall is reached
  3) precision is computed
  4) goto 2)
- typically a compromise



Boolean model

- set theory + Boolean algebra
- intuitive thinking
  - **document = set of terms**,
    hence, every document d_j represented as a binary vector
    of dimension **m** (each bit = appearance of a term in d_j)
- but
  - for querying, it is better to consider
    term = set of documents it appears in (the inverse)
  - hence, every term $\mathbf{t_i}$ could be represented as a binary vector
    of dimension **n** (each bit = t_i's appearance in a document)

Boolean model

- anyways, we have a binary **term-by-document matrix**
  - columns are document vectors
  - rows are term vectors
- based on properties of a text collection, we consider
  - a common document in the collection is not very long (1–100 text pages) → contains only fraction of the vocabulary (e.g., 1%) → the document vector is sparse
  - hence, the **term-by-document matrix is sparse**
    - note that also the term vectors must be sparse
    - the sparsity is typically huge, e.g., 99% zeros in the matrix

Boolean model

- consider the collection of Shakespeare's plays
- the term-by-document matrix would look like:

		document vector Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othelo	Macbeth
term vector	Antony	1	1	0	0	0	1
	Brutus	1	1	0	1	0	0
	Caesar	1	1	0	1	1	1
	Calpurnia	0	1	0	0	0	0
	Cleopatra	1	0	0	0	0	0
	mercy	1	0	1	1	1	1
	worser	1	0	1	1	1	0

the Shakespeare examples were borrowed from the slides created by **Alberto Simões**, who adapted those of **Heinrich Schütze**, University of Stuttgart

Boolean model – queries

- queries are Boolean expressions over terms
  - operations AND, OR, NOT
  - e.g., q = Brutus AND Caesar AND NOT Calpurnia
- easy model, **bit operations** over **term vectors**
  - e.g., Brutus = 110100, Caesar = 110111, Calpurnia = 010000
    q = 110100 AND 110111 AND NOT 010000
    = 100100
    $\Rightarrow$ Antony and Cleopatra, Hamlet

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othelo	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Antony and Cleopatra, Act III, Scene ii
Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius **Caesar** dead,
He cried almost to roaring; and he wept
When at Philippi he found **Brutus** slain.

Hamlet, Act III, Scene ii
Lord Polonius: I did enact Julius **Caesar** I was killed
in the Capitol; **Brutus** killed me.

Boolean model – inverted index

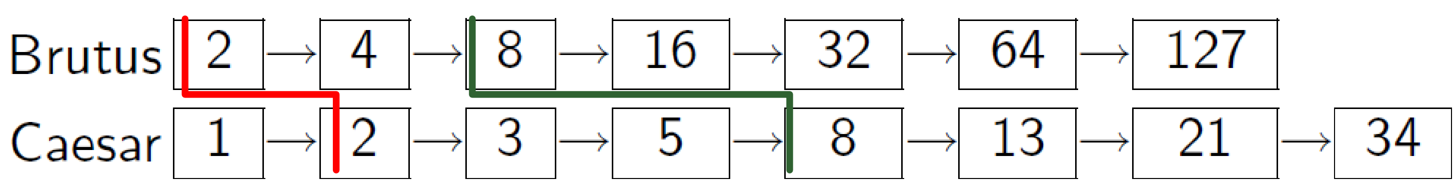
- the model is easy, but how to implement it?
  - the term vectors are huge
    - e.g., each million dimensions in case of collection of million documents
  - fortunately, the matrix is very sparse
- inverted index
  - compact representation of sparse matrix
  - each term vector is converted into an **inverted list**, consisting of **sorted** ids of documents it appears in

Brutus $\boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{8} \rightarrow \boxed{16} \rightarrow \boxed{32} \rightarrow \boxed{64} \rightarrow \boxed{127}$

Caesar $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{3} \rightarrow \boxed{5} \rightarrow \boxed{8} \rightarrow \boxed{13} \rightarrow \boxed{21} \rightarrow \boxed{34}$

Boolean model – inverted index

- the query is processed in a "merge sort" style
  - the cursors in individual lists ensure proper alignment,
    i.e., only the cursor pointing to the smallest id can move forward
- having the cursors aligned, the AND, OR, NOT operations are realized as set operations on sets of ids
  - AND = intersection, OR = union, NOT = complement
- example
  - q = Brutus AND Caesar
  - result = (2, 8)



Boolean model – optimizations

- store also frequency of each term in the collection
- what's better for fast processing ?
  - (a) Caesar AND (Brutus AND Calpurnia)

    or

    (b) (Caesar AND Brutus) AND Calpurnia

term	freq		lists							
Brutus	21	⇒	2	4	8	16	32	64	128	
Calpurnia	16	⇒	1	2	3	5	8	13	21	34
Caesar	3	⇒	13	16						

  - (b) is faster
    - because of smaller term frequency, the term Caesar AND Brutus results in smaller intermediate set, than Brutus AND Calpurnia does
    - results in less time/space cost

Boolean model – optimizations

- so far, we supposed a conjunctive query q = (AND … AND … AND …), which is quite easy to implement efficiently
- optimization of OR and NOT operations is more difficult
- in particular, q = (xxx OR yyy) AND (zzz OR www) could be optimized as
  - get frequencies for all terms
  - estimate the size of each OR by the sum of its frequencies (conservative)
  - process in increasing order of OR sizes

Boolean model – optimizations

- merging the vocabulary & the inverted lists into a single data structure (two tables)
  - vocabulary table is also a lookup table pointing to the lists (stored within another table)
- could be efficiently implemented and indexed in a traditional relational database

Term	#Docs	Col.F
ambitious	1	1
be	1	1
brutus	2	2
capitol	1	1
caesar	2	3
did	1	1
enact	1	1
hath	1	1
I	1	2
I'	1	1
it	1	1
julius	1	1
killed	1	2
let	1	1
me	1	1
noble	1	1
so	1	1
the	1	1
the	1	1
told	1	1
you	1	1
was	2	2
with	1	1
...	...	...

Doc #	TermF
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1
...	...

Boolean model – inverted index

- inverted index = **efficient** implementation of Boolean model, but, some conditions must be satisfied:
  - **frequent** terms **not indexed** (long inverted lists)
  - query consists of a **few keywords** (2-5)
    – multiplicative cost factor
  - the inverted lists are **sorted** (w.r.t. ids of documents)
  - the collection is rather **static**, otherwise, inserting new documents would require an expensive ids insertion, each into several (tens-hundreds) inverted lists

Boolean model – pros and cons

- pros
  - **easy model**, documents either match or mismatch a query
  - **efficient** implementation
  - professional searches (e.g. lawyers) still like Boolean queries (they **know what they search for**)

Boolean model – pros and cons

- cons
  - quite **rough model**, requiring the user to have **very precise intent** – otherwise too specific due to ANDs, or too general due to ORs
    - could be partially solved on a meta-level as: given just a set of keywords $k_1..k_j$, the query processor could try to query k_1 AND k_2 AND … AND k_j, if none/few results returned, then turn some ANDs to ORs, ending with k_1 OR k_2 OR … OR k_3
      - at the cost of less efficient queries
    - leads to worse **effectiveness** (precision and recall)
  - **no ranking** of the returned documents, all are the same relevant (how to navigate in the result?)

Extended Boolean model

- proposed by Salton et al in 1983
- motivation
  - to solve the cons of Boolean model – non-ranked results (leading to either too big or empty answer)
  - for example, consider two documents in a large collection
    - one about computers, one about mice
    - then a query: q = computer AND mouse,
      will not return any of the two documents,
      though some could be relevant (even both)

Extended Boolean model

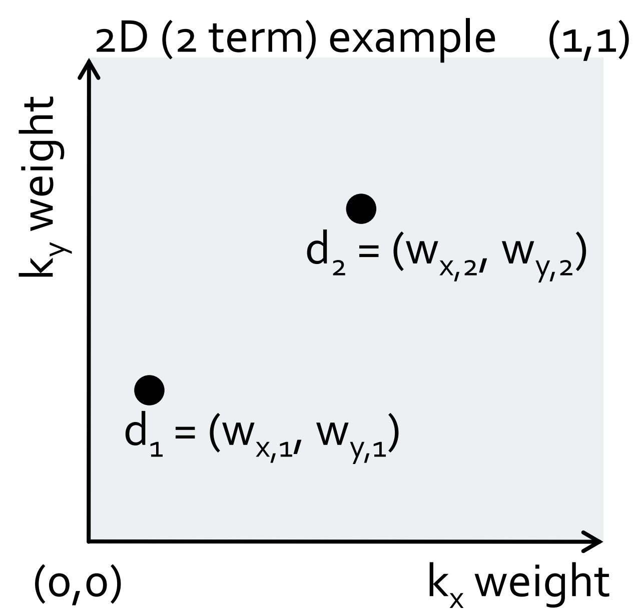
- the extension
  - weighted terms, allowing partial matching, thus introducing ordered results
  - i.e., the extended Boolean model offers to search
    - also **partially relevant** documents, instead of only
    - **fully relevant** as in the standard Boolean model
- combines Boolean algebra with Vector model

Extended Boolean model

- suppose a term k_x has a weight $w_{k,j}$ in a document d_j
  - $w_{k,j}$ says **how important** the term k_x is in document d_j
- let the weight be set as, e.g.,

$$w_{x,j} = f_{x,j} * \frac{idf_x}{\max_i idf_i}$$

- each document could be viewed as a vector of weights in **m**-dimensional space (**m** is the size of vocabulary)
- this part of the model is borrowed from the vector space model
  - more details on term weighting and on the vector space model in the next lecture



Extended Boolean model

- for a query q and each document vector d_j,
  the **relevancy score** of d_j to q is computed
  - the result is (logically) the whole collection ordered by decreasing relevancy scores
  - so, the user can choose where to stop browsing the result

- a query is evaluated by decomposing the formula q into
  - a conjunctive form φ_1 AND φ_2 AND … AND φ_t
  - or a disjunctive form φ_1 OR φ_2 OR … OR φ_t

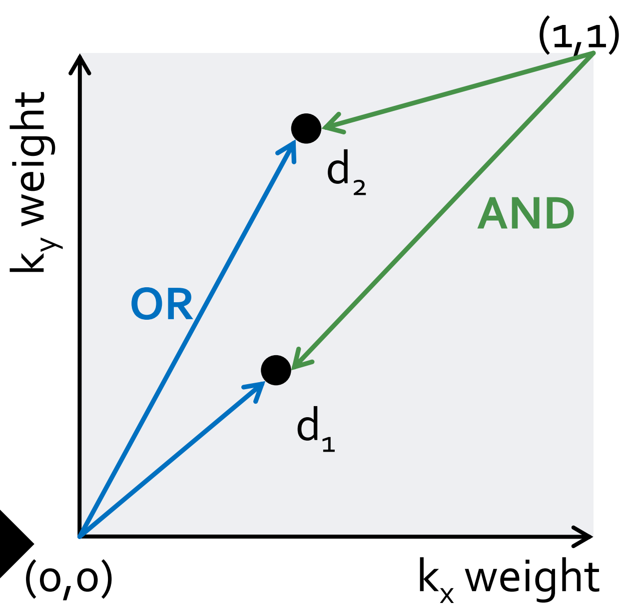
- the evaluation of ANDs and ORs is turned into measuring Euclidean distance in the space, while this is performed recursively

Extended Boolean model

- let's consider the two simple cases, where φ_i is directly a term k_i
  - q = k_1 AND k_2 AND … AND k_t
    the relevancy of a document d_j to q is computed as
    $$relev(q,d_j) = 1 - \sqrt{\frac{(1-w_{1,j})^2 + (1-w_{2,j})^2 \ldots + (1-w_{t,j})^2}{t}}$$
  - q = k_1 OR k_2 OR … OR k_t
    here the relevancy is computed as
    $$relev(q,d_j) = \sqrt{\frac{w_{1,j}^{\;2} + w_{2,j}^{\;2} \ldots + w_{t,j}^{\;2}}{t}}$$

note: in the formulas we only consider terms appearing in the query q

what's the intuition behind the geometry?



Extended Boolean model

- in other cases, when φ_i is subformula, the relevancy is computed recursively applying the same formulas as for ANDs and ORs with terms
  - the difference is that the $w_{i,j}$ components are just replaced by relev(q, φ_i)

- for example, the relevancy of a document d to the query

q = (k_1 AND k_2) OR k_3 is recursively evaluated as

$$relev(q, d) = \sqrt{\frac{\left(1 - \sqrt{\left(\frac{(1-w_1)^2 + (1-w_2)^2}{2}\right)}\right)^2 + w_3^2}{2}}$$

Extended Boolean model

- the relevancy computation could be even more generalized by use of so-called P-norms
  - new parameter $p \geq 1$
  - just replace the **powers of 2** and **square roots** by **powers of p** and **p-roots**

- the parameter p allows to tune the model to achieve better effectiveness (quality of retrieval)

- for the previous example, we get

$$relev(q,d) = \sqrt[p]{\frac{(1-\sqrt[p]{(\frac{(1-w_1)^p+(1-w_2)^p}{2})})^p + w_3^p}{2}}$$

Extended Boolean model – pros and cons

- pros
  - still easy for the users – the same query expressions
  - ranked queries, the user can specify how large the answer should be
  - much better effectiveness over the standard Boolean model (the quality of retrieval)

- cons
  - computationally expensive due to
    - the p-powers and p-roots computations
    - maintenance of the ordered (intermediate) results