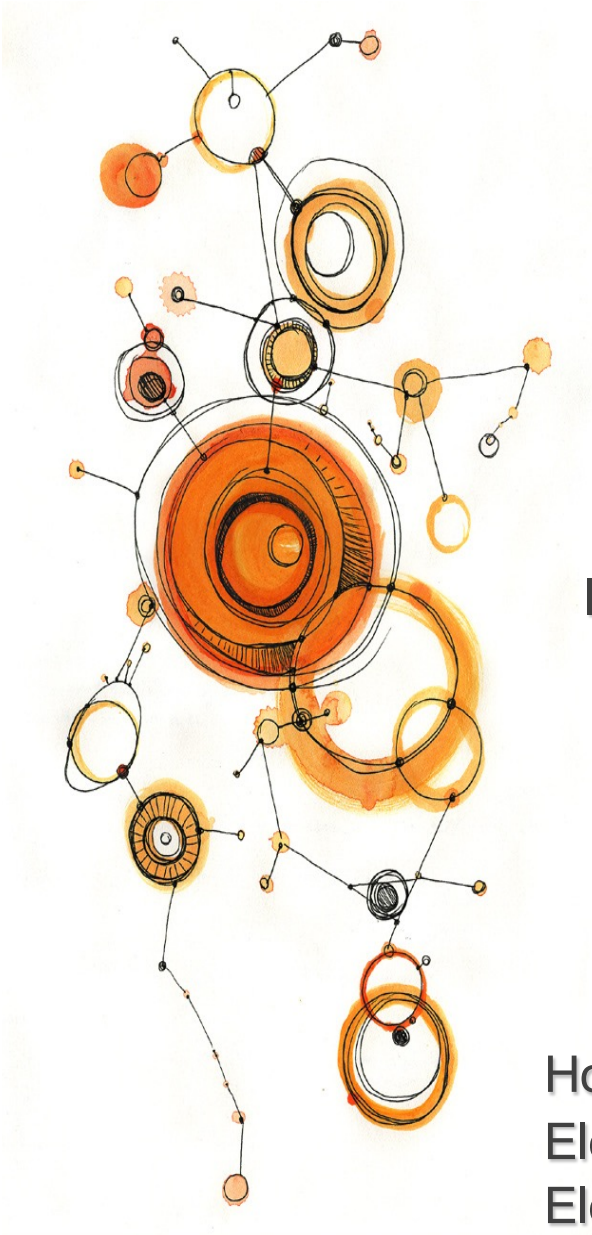


Modeling and Processing of Multimedia Data

International Second cycle degree programme (LM) in
Digital Humanities and Digital Knowledge (DHDK)
University of Bologna

Textual Information Retrieval Systems – Part I

Home page: http://www-db.disi.unibo.it/courses/DMMMDB/
Electronic version: 1.01.TextualInformationRetrieval-I.pdf
Electronic version: 1.01.TextualInformationRetrieval-I-2p.pdf

Outline

- Information Retrieval (IR): an introduction
- Textual documents representation in IR systems
- Automatic indexing techniques
- Searches of Boolean type
  - Basics on Boolean algebra and truth tables
- Searches of phrases and for proximity

Information Retrieval motivation (1)

- Information Retrieval (IR) deals with the *representation*, *storage*, *organization*, and *access* to information items
- The *representation* and *organization* of the information items should provide the user with easy access to the information in which she is interested
- Unfortunately, the *characterization of the user information need is not a simple problem*!! ☹

Information Retrieval motivation (2)

- Example of user information need in the context of a *Cultural Digital Library*:

  "*Find all documents containing information on Renaissance painters that (1) worked for a Pope and (2) were born outside of Italy. To be relevant the documents must include information about the locations of the artist's most famous paintings and the number artist's painting located in Rome*"

- Clearly this full description of user need cannot be used directly to retrieve information using current technologies
- Instead, the user must *translate* her information need into a *query* which can be processed by an IR system
  - Commonly the translation yields a set of *keywords* (or *index terms*) which summarize the description of the user information need

Information Retrieval goal

- The main task of an IR system is:

- Given a query, which represents the "information needs" of the user, and a collection of documents
- Retrieve the documents in the collection that are "relevant" to the query, returning them to the user in decreasing order of relevance

- Document collections are modeled as **unstructured data**, i.e., data without a schema (default internal organization) able to describe them or to assign a specific semantic

Relevant examples of IR applications

- Web search engines are the most visible IR application (e.g., Bing, Google, Yahoo, etc.)
- Many universities and public libraries use IR systems to provide access to book, journals, and other documents
- Among further relevant examples:
  - Social networks applications like Twitter, Facebook, Flickr, …;
  - Web digital libraries such as Wikipedia, CiteSeer, Google Scholar, …;
  - Multimedia digital libraries;
  - The *Europeana* initiative, and the related *European Digital Library* for the access to cultural digitized objects like books, photos, maps, audio, movies, and archival records from Europe's libraries, archives, museum and audio-visual collections
  - …

Study path

- We focus on **unstructured documents** representation, their indexing techniques, and models of queries
  - starting from traditional **textual documents** and then
  - continuing with the most complex **multimedia documents** managed by Multimedia Information Retrieval (MIR) systems

Textual document representation

- Textual documents are usually represented as bags (i.e., multi-sets) of representative keywords called "index terms"
  - following the *Bag of Words* model
- Index terms are used to summarize the document content
- An index term can be:
  - a keyword, chosen from a group of selected words (usually nouns)
    - This approach is particularly useful to *classify* documents, although it requires a manual intervention
  - any word, also known as *full-text indexing*
- Complex index terms may also be defined, such as groups of nouns (e.g., computer science)

- Alternatively, the composing terms are treated separately and the group is reconstructed by looking at the positions of the words in the text

Query representation

- Queries follow a similar approach
  - i.e. a query is a “set of words”

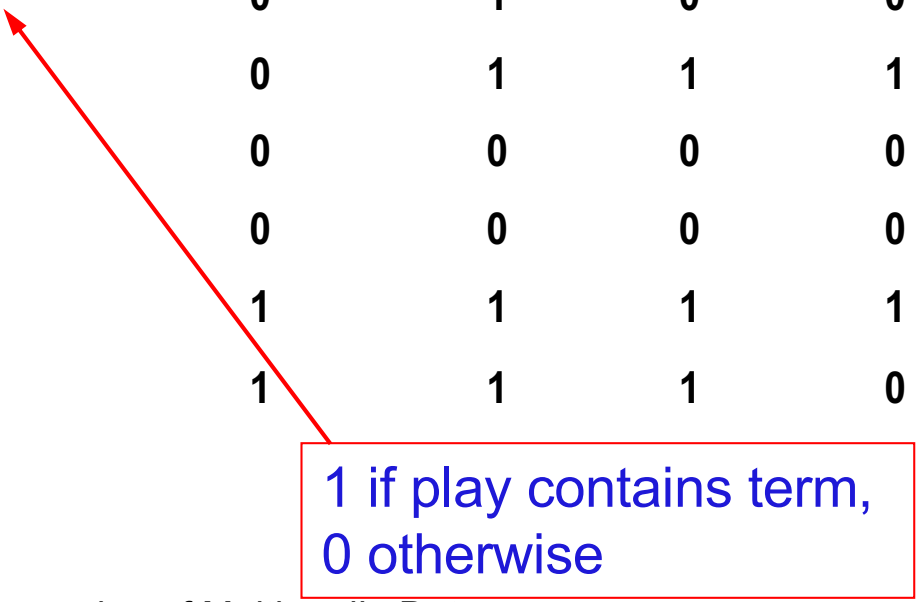
- However, how query terms are combined is an issue…

Boolean queries

- The simplest retrieval model is based on Boolean algebra:

Which plays of Shakespeare contain the words
Brutus AND **Caesar** AND NOT **Calpurnia**?

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	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



Recall on truth tables

- A truth table is a mathematical table used in logic to compute the functional values of logical expressions on each of their functional arguments
- Truth tables can be used to tell whether a propositional expression is true for all legitimate input values, that is, logically valid
- Among the principle operators:
  - **NOT**, **AND**, and **OR**

NOT	
V	F
F	V

AND	V	F
V	V	F
F	F	F

OR	V	F
V	V	V
F	V	F

Computing the results

- For each term we have a binary vector, with size
  N = number of documents in the collection
- Bit-wise Boolean operations are enough to compute the result:

  Brutus = (110100), Caesar = (110111), Calpurnia = (010000)

  (110100) AND (110111) AND NOT (010000) = **100100**

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	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
Result =	**1**	**0**	**0**	**1**	**0**	**0**

Is the matrix solution a good idea?

- Assume we have a collection of N = 1M documents
- Also assume that the overall number of distinct terms is V = 100K, with each document containing, on the average, 1000 distinct terms
- The matrix consists of

  100K x 1M = 10^{11} = 100G boolean values, with only 1% (1G) of 1's

- Space overhead suggests to look for a more effective representation
- Further, consider taking bit-wise AND and OR over vectors of 1M bits…

- The commonest solution adopted in text retrieval system is a structure known as "inverted index" (also: "inverted file")
- There are many variants of the inverted index, aiming to:
  - Support different query types
  - Reducing space overhead

Building the inverted index (1)

1) Documents are parsed to extract terms…

doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

2) Terms are sorted…

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Building the inverted index (2)

3) Multiple occurrences of a term in the same document are merged and frequency information is added…

Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

4) The index is then split into a "dictionary/vocabulary" and a "posting file"

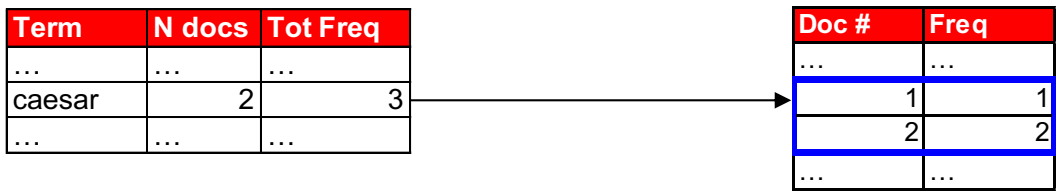
Term	N docs	Tot Freq
ambitious	1	1
be	1	1
brutus	2	2
capitol	1	1
caesar	2	3
did	1	1
enact	1	1
hath	1	1
I	1	2
i'	1	1
it	1	1
julius	1	1
killed	1	2
let	1	1
me	1	1
noble	1	1
so	1	1
the	2	2
told	1	1
you	1	1
was	2	2
with	1	1

Doc #	Freq
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

Inverted index size

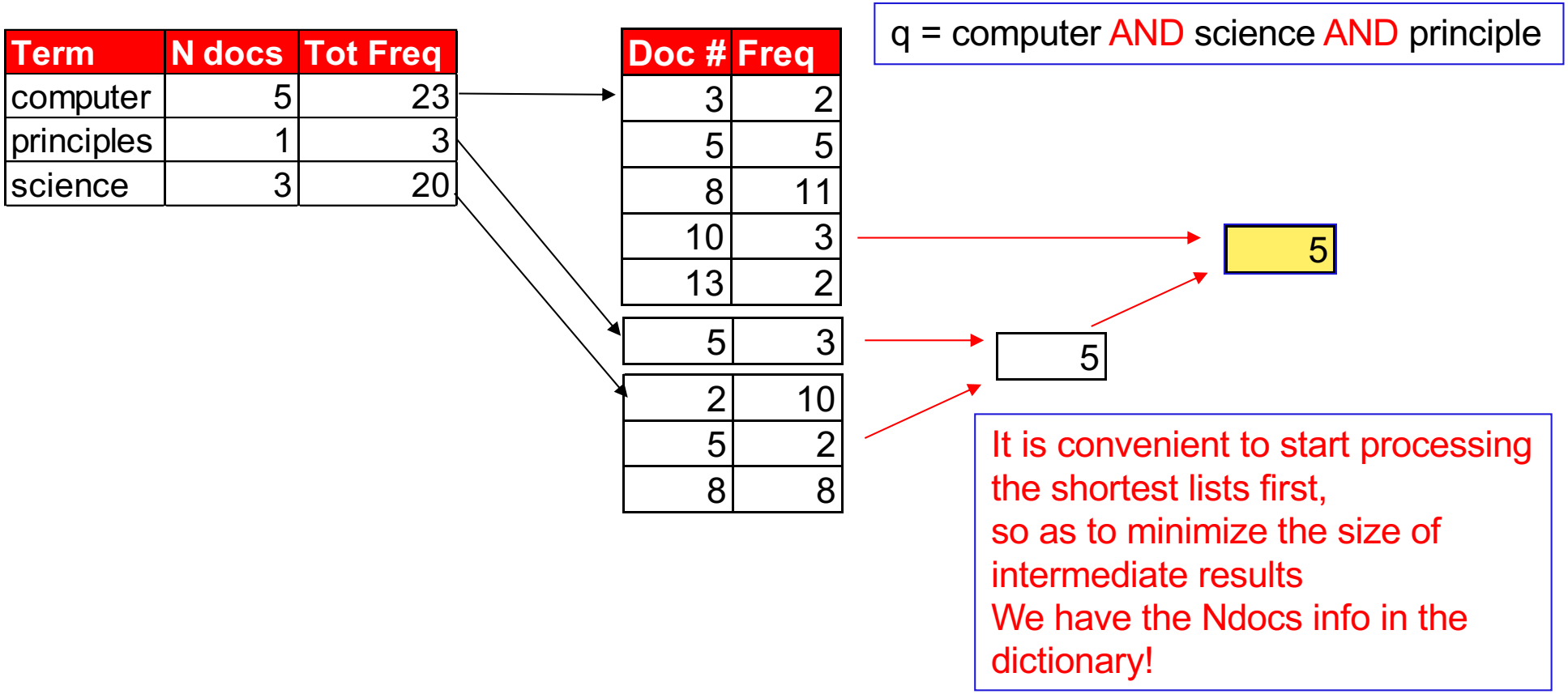
- Consider the size of the
  - Dictionary: with 100K terms, even assuming that a vocabulary entry requires 30 bytes on the average, we need just 3MBytes
  - Posting file: if each of the 1M documents contains about 1000 distinct terms, we have 1G entries in the posting file, each of them referenced by a distinct pointer

- A more effective space utilization is obtained by means of *posting lists*:
  - For each distinct term, have just one pointer to a list in the posting file
  - This "posting list" contains the id's of documents for that term and is ordered by increasing values of documents identifiers
  - Continuing with the example, this way we save 1G – 100K pointers!
  - Techniques are also available to "compress" the info within each list



Using the inverted index with Boolean q.'s

- ANDing two terms is equivalent to intersect their posting lists
- ORring two terms is equivalent to union their posting lists
- t1 AND NOT(t2) is equivalent to look for doc id's that are in the posting list of term t1 but not in that of t2



What to index?

- Most common words, like “the”, “a”, etc., takes a lot of space since they tend to be present in all the documents
- At the same time, they provide little or no information at all
  - However, what about searching for “to be or not to be”?
- Such words are frequently referred as stopwords
- A (language-specific) stopword list can be used to filter out those words that are not to be indexed
- The “rule of 30”:

  ~30 most common words account for ~30% of all tokens in written text
  - By eliminating the 150 most common words from indexing cuts 25% to 30% of space for postings

Remark: in practice, things are more complex, since we may want to deal with:

- Punctuation: State-of-the-art, U.S.A. vs. USA, a.out, etc.
- Numbers: 3/12/9, Mar. 12, 1991, B-52, 100.2.86.144, etc.
- …

Stemming

- In order to save space and to improve the chance of retrieving a document, a process called “stemming” is usually executed before indexing, so as to reduce terms to their “roots”
  - e.g., automate(s), automatic, automation all reduced to automat

for example compressed and compression are both accepted as equivalent to compress.	for exampl compres and compres are both accept as equival to compres.

- It is experimentally shown that stemming can reduce the number of terms by ~40%, and total index size by ~30%

- For details on how stemmers (= stemming algorithms) operate:

  `http://www.comp.lancs.ac.uk/computing/research/stemming/general/index.htm`

Thesauri

- Synonyms can be viewed as equivalent terms
  - E.g., car = automobile
- A text IR system usually comes with a thesaurus that, in its simplest form, consists of:
  1. A list of (important) terms
  2. For each term, a set of related words
- Related term = synonyms, hypernyms (car is a kind of…), hyponyms (… is a kind of car), etc.
- Actually, a thesaurus constitutes a *semantic network* of terms

- For real examples, please take a look at:
  - Wordnet (www.cogsci.princeton.edu/~wn/)
  - Merriam-Webster thesaurus (www.m-w.com)

The WordNet interface

Overview for "car"

The **noun** "car" has 5 senses in WordNet.

1. **car**, auto, automobile, machine, motorcar -- (4-wheeled motor vehicle; usually propelled by an internal combustion engine; "he needs a car to get to work")
2. **car**, railcar, railway car, railroad car -- (a wheeled vehicle adapted to the rails of railroad; "three cars had jumped the rails")
3. cable car, **car** -- (a conveyance for passengers or freight on a cable railway; "they took a cable car to the top of the mountain")
4. **car**, gondola -- (car suspended from an airship and carrying personnel and cargo and power plant)
5. **car**, elevator car -- (where passengers ride up and down; "the car was on the top floor")

Search for Synonyms, ordered by estimated frequency of senses

☑ Show glosses

☐ Show contextual help

Search

Results for "Hyponyms (...is a kind of this), brief" search of noun "car"

Sense 1
car, auto, automobile, machine, motorcar -- (4-wheeled motor vehicle; usually propelled by an internal combustion e
=> ambulance -- (a vehicle that takes people to and from hospitals)
=> beach wagon, station wagon, wagon, beach waggon, station waggon, waggon -- (a car that has a long body
=> bus, jalopy, heap -- (a car that is old and unreliable; "the fenders had fallen off that old bus")
=> cab, hack, taxi, taxicab -- (a car driven by a person whose job is to take passengers where they want to go
=> compact, compact car -- (a small and economical car)
=> convertible -- (a car that has top that can be folded or removed)
=> coupe -- (a car with two doors and front seats and a luggage compartment)
=> cruiser, police cruiser, patrol car, police car, prowl car, squad car --
(a car in which policemen cruise the streets; equipped with radiotelephonic communications to headquarters)
=> electric, electric automobile, electric car -- (a car that is powered by electricity)
=> gas guzzler -- (a car with relatively low fuel efficiency)
=> hardtop -- (a car that resembles a convertible but has a fixed rigid top)
=> hatchback -- (a car having a hatchback door)

Using thesaurus' information

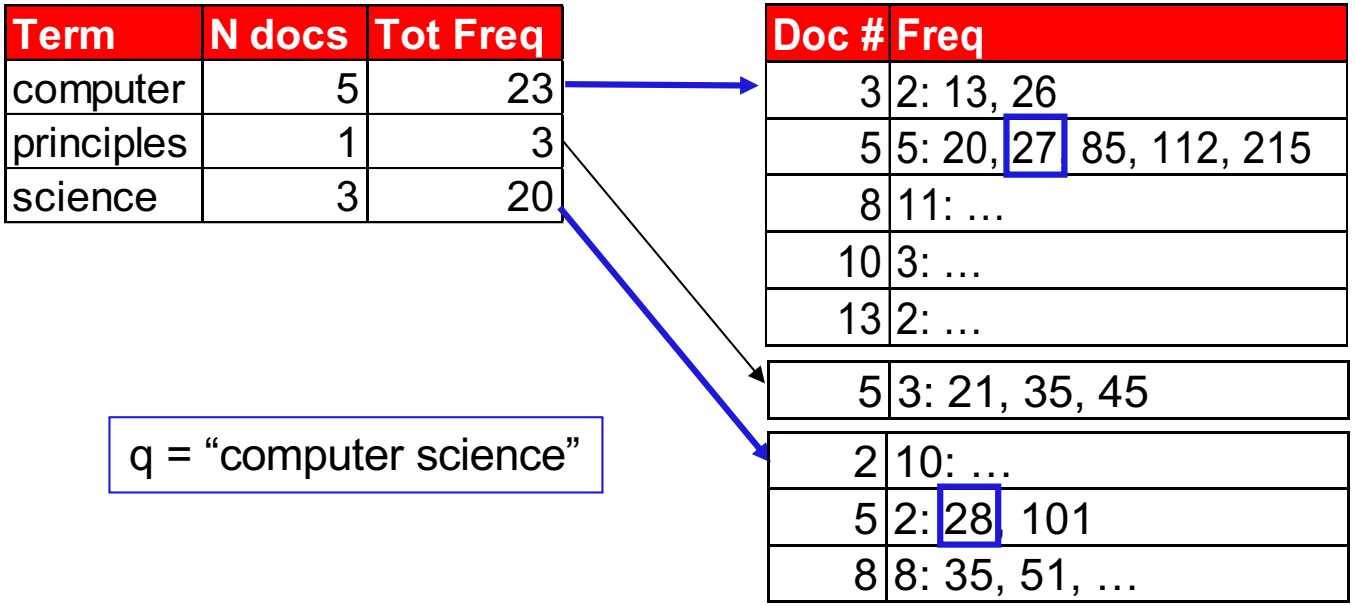
- If your query includes the term "car", there are two possibilities to exploit thesaurus' information:
  1. The system can "expand" the query

     E.g., car AND tyres becomes (car OR automobile) AND tyres
  2. The system can build the inverted index by also placing docs containing the term "car" in the posting list of "automobile", and vice versa

- Usually query expansion is preferred, in order to avoid excessive index growth
- However, query expansion slows down query processing…

Phrase and proximity queries

- Searching for phrases (e.g., “the challenge of information retrieval”) can be implemented by extending the entries in the posting file with positional information (= position of the term in the document)



- Proximity queries are a relaxed version of phrase queries, where we just require that the query terms are “close” each other

  E.g., Gates NEAR Microsoft

Limits of the Boolean retrieval model

- Although the Boolean model has a clear semantics and is also appreciated by experienced users, it has several drawbacks:

1. Unexperienced users have difficulties in understanding what Boolean operators really mean
2. No notion of "partial matching"
3. No ranking of the documents
4. "Near-miss" and "Information overload" problems

- Several extensions of the basic Boolean model have been proposed, in order to solve above problems
- We skip them, and directly move to consider the "vector space model"…