

# Chap. 1: Information retrieval basics

Jean-Pierre Chevallet

LIG-MRIM

Sept 2016

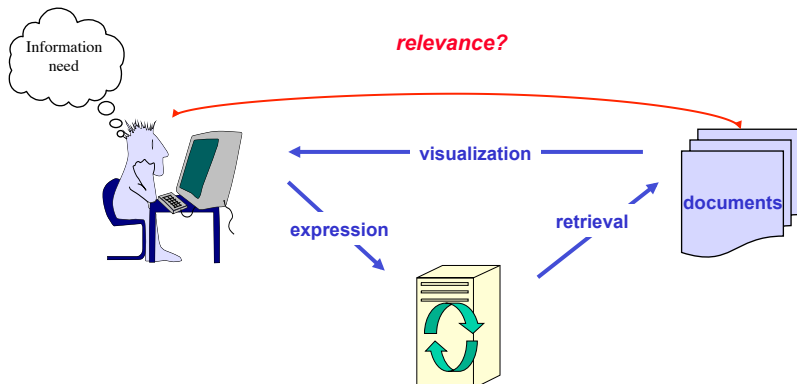
# Outline

- 1 Key notions in IR
- 2 Relevance
- 3 IR Context
- 4 Anatomy of a text search engine

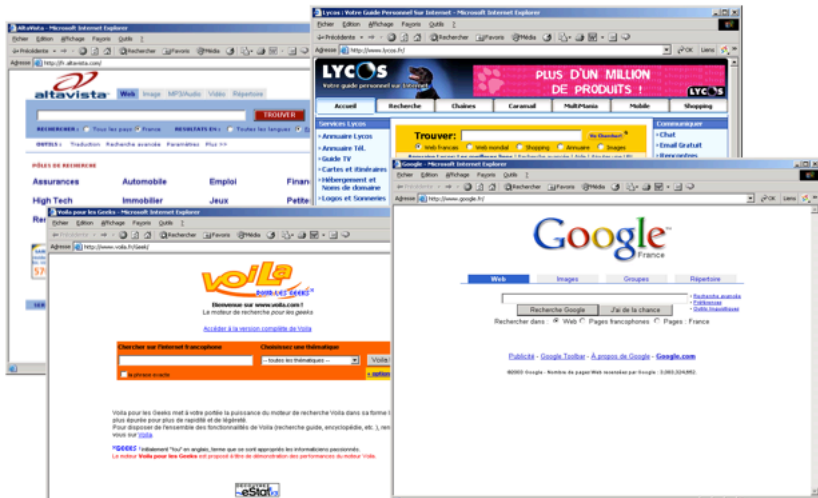
# Challenge of IR

Challenge of Information Retrieval:

Content base access to documents that satisfy an user's information needs



# Typical use: search Engine in the WEB



## But also .. Mobile Information Access



# Important Notions

- IR definition:
  - "Information retrieval (IR) deals with the representation, storage, organization of and access to information items" [Baeza-Yates and Ribeiro-Neto, 1999],
  - "Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)" [Manning et al., 2008]
- What about:
  - Information ?
  - Document ?
  - User need ?

# Important Notions

- Central elements for IR:
  - Documents
  - Document content
  - Information need of a user
  - Satisfaction of the user
- Information:  
Is what a user gets from documents using his own knowledge

# Information

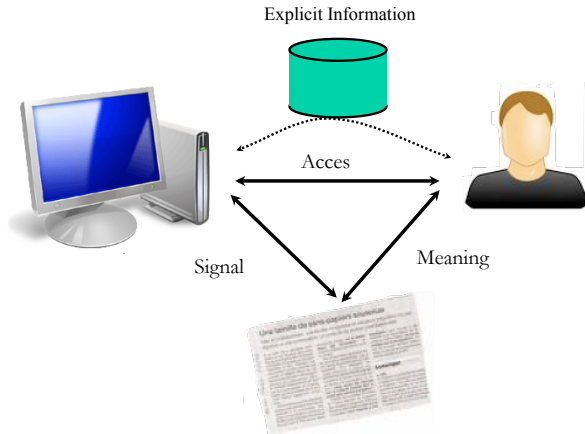
- Which kind of information ?
- transform a trace into information (ex: fire place)



# Role of IRS

- The role of an IRS:  
An automatic mediator between user and documents
- How to match user need and document ?  
Express user need into a query  
Can we compute a match between query and document  
without external informations ?

# Relevance



## Multimedia queries

- Show me x-ray images with fractures of the femur.
- Zeige mir Röntgenbilder mit Brüchen des Oberschenkelknochens
- Montre-moi des fractures du fémur.



# Documents

Document as a form:

## Media

- Text, still image, video, structured documents

## Type

- Text : book, article, letter, ...
- Image : X-Rays, Photographs, Graphics,

## Granularity and structure

- Text : whole document, structure element (chapter , section, paragraph, sentence), passage (window of x words in a text), notion of **doxel** as documents atoms.
- Video : whole video, a shot, an image of the video

## Documents aspects

- Physical (form):** An object material or not, a proof function, an information support, a structure, digital = need a tool to be read
- Meaning (content):** A sign with a meaning and an intension, context is part of the meaning construction
- Social (medium):** A medium for social relationship, a trace, constructed or found, of a communication that exists outside space and time; at the same time, it is an element of identity systems and a vector of power.

# Documents

From : "Document: Form, Sign and Medium, As Reformulated for Electronic Document" Roger T. Pedauque, STIC-CNRS

- An electronic document is a data set organized in a stable structure associated with formatting rules to allow it to be read both by its designer and its readers.
- An electronic document is a text whose elements can potentially be analyzed by a knowledge system in view of its exploitation by a competent reader.
- An electronic document is a trace of social relations reconstructed by computer systems

# Document in IR

**As an object:** the item returned as an answer. Still clear in the digital word ?

**As a sign:** the content that interests the reader, the aspect to be analyzed and indexed. A sign is supposed to have a meaning, and it is this meaning that counts for an IRS user.

**As a medium:** for social relationship, a trace, also used for collaborative works

# Document content

2 classes of information

**As an object:** Meta-Information (information about the document)

Attributes: title, author, creation date, etc.

Structure (content organization): logical and physical structures, links, etc.

**As a sign:** Content

Raw content : the initial document

Semantic content: extracted information from the raw content



# Outline

- 1 Key notions in IR
- 2 Relevance**
- 3 IR Context
- 4 Anatomy of a text search engine

# User Query

User's information need

Use of queries according to a predefined language

- Constrains on meta-information

Attributes : **novel written by Victor Hugo**

attribute on document type and author

Structure : **article on football containing a photograph**

Structure of links between text and image

- Constrains on the content

Raw content : **letter with the text "I came, I saw, I conquered"**

Retrieval on character strings

Semantic content : **documents about information retrieval,**  
retrieval symbolic descriptions

# Satisfaction of the user

Some criteria for user satisfaction:

- The system should be simple to use
- The system must give the best possible answers, and these answers must be relevant to the user
  - System relevance versus user relevance
  - Granularity of relevant information
- The system must return "reasonable" quantities of answers
- The system must give fast answers

Very hard to satisfy all these points

## User's need

Taking into account of the expertise of a user

### Domain expertise of the user

One information need expressed the same way by two persons should not necessarily give the same answers.

## User's need

Taking into account of the external context of the information need

### Temporal

One information need expressed at two different moments does not give the same answers: "tsunami" at the end of december 2004 is "obviously" related to what append in Asia.

### Geographical

One information need expressed at two different places does not have the same meaning: "restaurant" in Grenoble do not necessarily need to give restaurants of New-York in the answer.

## Relevance at Document Side (Mizzaro 97 [Mizzaro, 1997])

### Document

The physical entity that the user of an IRS will obtain after his seeking of information.

### Surrogate

A representation of a document.

### Information

What the user receives when reading a document.

# Relevance at User Side

## Information need

A representation of the problem in the mind of the user.

## Request

A representation of the information need of the user in a "human" language, usually in natural language.

## Query

A representation of the information need in a "system" language, for instance Boolean.

# Relevance: definition

Relation between:

## Information Source

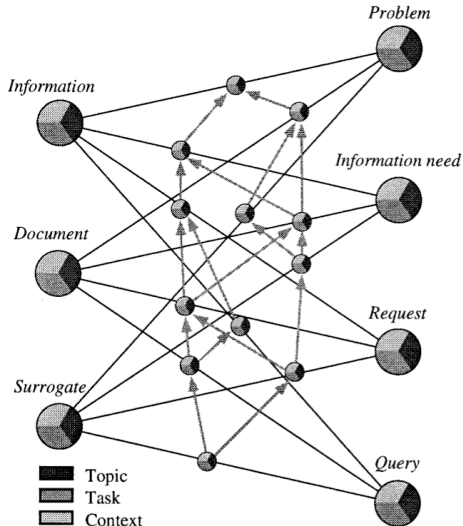
Document  
Surrogate  
Information

## Information Target

Information need  
Request  
Query



# Relevance Relations (Mizzaro)



From Mizzaro 97

# Outline

- 1 Key notions in IR
- 2 Relevance
- 3 IR Context**
- 4 Anatomy of a text search engine

# IR Context

IR context can be relate to:

## Cognitive state of the user

IR context > Seeking context > User task context  
Interactive searching  
Personalisation in search

## Physical state of the user+device

Physical context as measured by sensors

# IR Context

Tacking into account the context

To enhance precision of access (classical IR)

Filter answer using context.

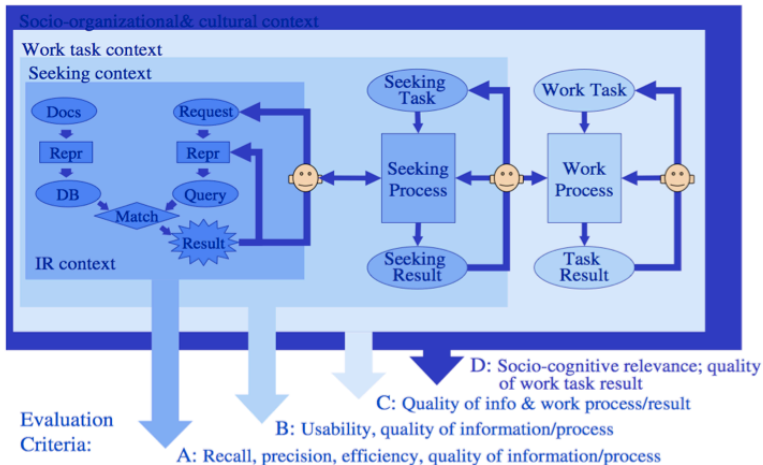
For more interactivity (filtering IR)

System can "push" information triggered by context

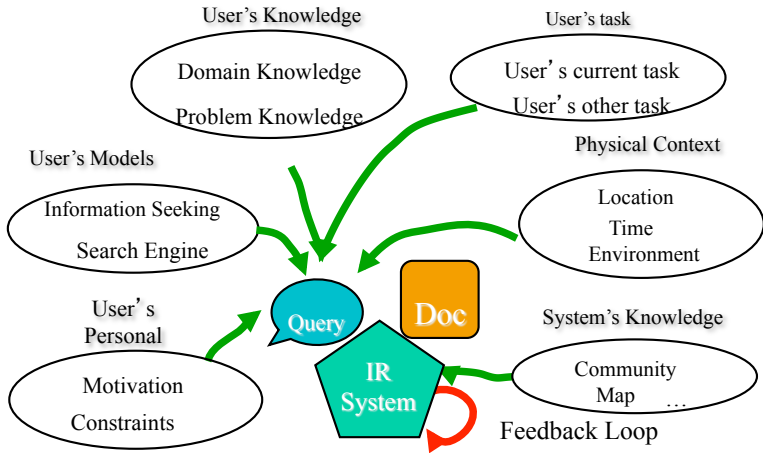
## Context layers

- IR context: Related to only one query, against the IRS
- Seeking context: A session of several queries related to the same seeking task
- Work task context Several sessions of information seeking for a given task
- Social and cultural context What the task is used for ?

# Context



# Context



# Outline

- 1 Key notions in IR
- 2 Relevance
- 3 IR Context
- 4 Anatomy of a text search engine**



# Anatomy of a text search engine

Two main tasks:

**Indexing:** build a structure (index) to quickly access document by content

**Querying:** use this structure to solve a query.

# Indexing

Extract part of documents elements:

- Possible initial *annotation step*: associate entities (concepts) to documents.
  - Need dedicated resources: thesaurus, lexical base, etc
  - Need automated NLP annotation process
  - Is the definition of semantic indexing
- Otherwise: extract text "chunks" called token

# Tokenization

*tokenization is the task of chopping it up into pieces, called tokens , perhaps at the same time throwing away certain characters, such as punctuation<sup>1</sup>*

*A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing.*

---

<sup>1</sup>Introduction to Information Retrieval By Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze

## Tokenization: example

**Input:** " Friends, Romans, Countrymen, lend me your ears;"

**Output:** Friends Romans Countrymen lend me your ears

# Tokenization: how ?

Define tokens by a context free grammar, or by a regular language.

## Context free grammar

$G = (V, \Sigma, R, S)$ :

$V$  finite set of non terminal symbols

$\Sigma$  finite set of terminals, different from  $V$ . The alphabet of the language.

$R$  finite set of production rules, as relation between  $V$  and  $(V \cup \Sigma)^*$ , note that this can be empty noted  $\epsilon$

$S$  a starting symbol from  $V$

Rule application: by simple rewriting.

## Context free grammar exemple

$$V = \{N, S\}$$

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .\}$$

$$R =$$

$$F \rightarrow 0|1|2|3|4|5|6|7|8|9$$

$$N \rightarrow F$$

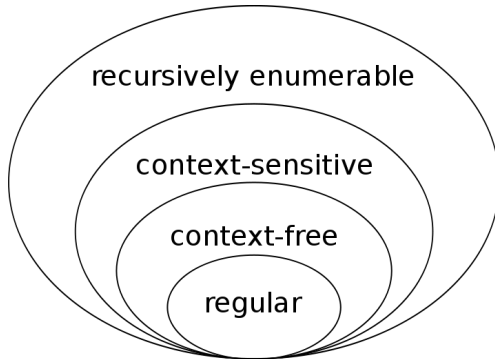
$$N \rightarrow NN$$

$$S \rightarrow N$$

$$S \rightarrow N.N$$

## Grammar order

Noam Chomsky hierarchy of formal grammar:





## Regular language

A regular language is a context free grammar.

Recursive definition:

- $\emptyset, \epsilon$  an empty set, and the empty string are regular.
- $\Sigma$  all simple vocabulary elements are regular
- $\cup$  union of regular languages are regular
- concatenation of regular languages are regular
- \* zero or more concatenation of regular languages are regular

Easy to write an analyser: an automata.

The previous example is regular (left to demonstration).

## Tokenization: problems

**Input:** "Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing."

Possible tokenizations:

Neill, neill, ONeill, oneill, o'neill, o neill, ...  
aren t, arent, are n't, ...

# Tokenization: solutions

- Depends on the corpus language and domain and indexing objectives.
- Raise complexity rules to cope with more situation:
  - syntax of dates
  - syntax of email addresses
  - list of special names like C++
  - ...

## Indexing: generating (token,doc id)

- Tokenization of text
- Can drop some useless token : *stop words*
- Can normalize token
  - Groupe different token in the same class
    - removing some caps
    - treating accents and diacritics
    - grouping token in terms (or words)
    - special treatments for acronyms
    - etc ...
- Produce a sequence of couple (token, doc id)
- Also to token can be transformed into an internal id, using a *dictionary data structure*
- token after treatment becomes *index terms*

## Indexing: exemple

D1 "Addis Abeba is a nice city"

D2 "Nice attacker grew beard in week before"

D3 "John Beard lives in city of Addis, in Louisiana"

Tokenisation:

- Stop words: is, a , in , the, of, before
- Remove caps and diacritics

# Inverted index

Also known as *posting file*

- A smart structure to retrieve quickly all documents that contain an index term
- Each index term points to the ordered list of document where its appears.
- Inverted index buildings steps:
  - ① extraction of couple (term, doc id)
  - ② sort the list on term then on doc id
  - ③ group terms and build inverted list (or posting list)

## Extraction phase

(addis,1) (abeba,1) (nice,1) (city,1) (nice,2) (attacker,2) (grew,2)  
(beard,2) (week,2) (john,3) (beard,3) (city,3) (addis,3)  
(louisiana,3)

## Sorting phase

(abeba,1) (addis,1) (addis,3) (attacker,2) (beard,2) (beard,3)  
(city,1) (city,3) (grew,2) (john,3) (louisiana,3) (nice,1) (nice,2)  
(week,2)



## Build the inverted data structure

abeba	→	1
addis	→	1,3
attacker	→	2
beard	→	2,3
city	→	1,3
grew	→	2
john	→	3
louisiana	→	3
nice	→	1,2
week	→	2

## Put the inverted structure in a file

Simply a file composed of sequence of doc id:

- the term point to the start of the sequence
- store also the length of the sequence (no know where to end).

File = 

1	1	3	2	2	3	1	3	2	3	3	1	2	2
---	---	---	---	---	---	---	---	---	---	---	---	---	---

One can leave some "room" for dynamic adding to the index file.

## Put the inverted structure in a file

File = 

1	1	3	2	2	3	1	3	2	3	3	1	2	2
---	---	---	---	---	---	---	---	---	---	---	---	---	---

Table: Inverted index term entry

Indexing term	length	pointer
abeba	1	0
addis	2	1
attacker	1	3
beard	2	4
city	2	6
grew	1	8
john	1	9
louisiana	1	10
nice	2	11
week	1	13

# Indexing: how to ?

- large collection:
  - Block sort-based indexing:
    - Build the couple list in block memory of a fix size (buffer)
    - When buffer is full (or at end of collection): sort and store the buffer on disk
    - Finely read all the sorted buffer in parallel in order to build the index file.
- Dynamic indexing:
  - leave "room" for the increase of the list
  - start a new index and fuse later

# Querying

- Treat query like document (tokenize, etc ..)
- Open the index, and combine the document lists

Weighting ? ... next course.

## Posting list questions

- 1 What is the interest of sorting the posting list ?
- 2 How to quickly access to one list having a term ?
- 3 Where is stored document frequency ?
- 4 What is the relation between length and pointer ?
- 5 How to compress the posting file ?
- 6 Where to store complementary information like term frequency ?

## Question: corpus and index size

Suppose that a english text is composed of words which have on average 5 characters, and for each word in the text there is one space and one diacritic. Suppose also that characters are coded in ASCII, so using one byte per character. Suppose that you have a collection of  $C$  document with an average size of  $N$  different words.

- 1 If each word appears only once in document, what is the size of this collection in byte ?
- 2 If the corresponding posting file codes doc-id using 32 bits, what is the size of this file ?
- 3 Suppose now that we have a lot of small documents, so that  $C$  become larger than 32bits, and we use 64 bit integer to store docId. What is the evolution of the size of the index file compared to the corpus size ?

## Posting list answers





- 1 To faster list intersection when querying.
- 2 One must have a lexicon (or dictionary) with fast access from word index like trees or hash tables.
- 3 This is the size of the posting list for each term.
- 4 This is the sum of the size from beginning till position of the term, so the pointer can be deduced from size are is redundant.
- 5 By reducing the number of byte allocated to document Id, if there is less that  $2^{32}$  document, also by storing only difference values between two successive values that are always increasing: so just gap between docId are stored.
- 6 In the posting list, with the doc ID.



## Answer: corpus and index size

- 1 If each word appears once in the document, then there is exactly  $N$  words per document, so  $N \times C$  in the collection. As each word is on 6 byte, the size is  $N \times C \times 6$
- 2 Each posting uses 4 bytes. Because each word is unique in each document, each word in each document will be associated to a docid in the posting list. So the size of the posting list is  $N \times C \times 4$
- 3 If  $N$  grow and we use 64 bits doc Id then the index size becomes  $N \times C \times 8$  as the corpus size still is  $N \times C \times 6$ , the index is now much bigger that the corpus itself !

# References

-  Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999).  
*Modern Information Retrieval*.  
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
-  Croft, B., Metzler, D., and Strohman, T. (2009).  
*Search Engines: Information Retrieval in Practice*.  
Addison-Wesley Publishing Company, USA, 1st edition.
-  Manning, C. D., Raghavan, P., and Schütze, H. (2008).  
*Introduction to Information Retrieval*.  
Cambridge University Press, New York, NY, USA.
-  Mizzaro, S. (1997).  
Relevance: The Whole History.  
*Journal of the American Society of Information Science*, 48(9):810–832.