# Introduction to Information Retrieval

## Chapter 1. Basic Concepts in IR

***Dr. SAID KADRI***

**Associate Professor**

**Department of Computer Science, Faculty of Mathematics and Informatics,**

**University Mohamed Boudiaf of M'sila**

**E-mail:** kadri.said28@gmail.com

**Website:** https://kadrisaid28.wixsite.com/sgadri

# Table of content

Web Search Engine

Web Search Engines - problems

Web Search as a huge IR system

Web Search Systems
Matching a query to a document

Processing Boolean queries

Different search engines

Web Search

Background and history

Early web search engines and web collections

Indexing the web

Web as a graph

Spam in IR

Different types of web search users

User query needs

Why is Web Information Retrieval Difficult?

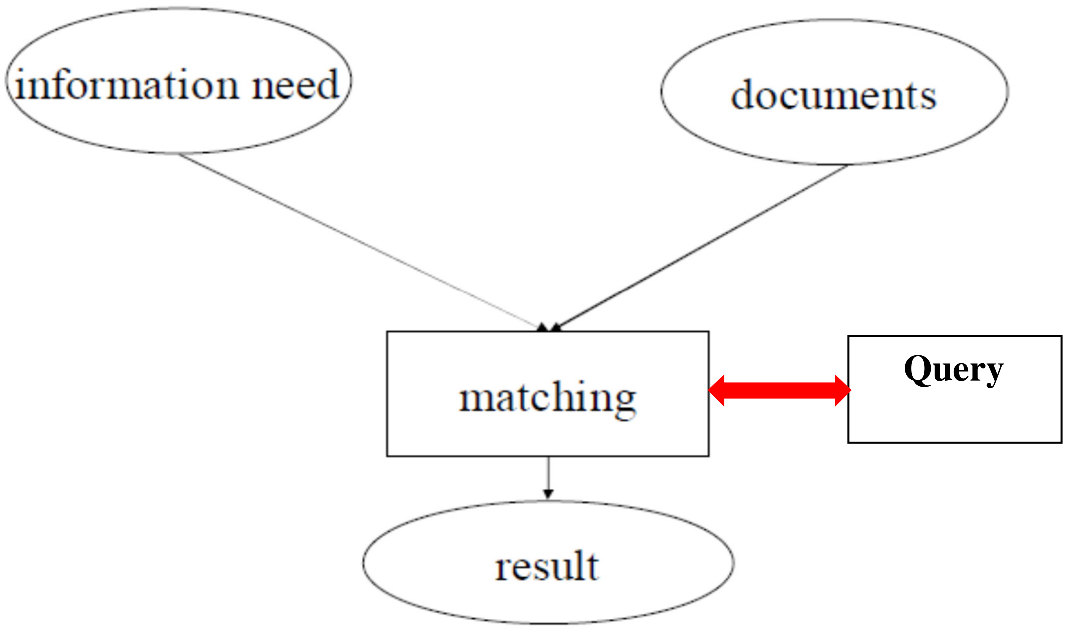
# Introduction to Information Retrieval

## Introduction

- Text mining refers to data mining using text documents as data.
- Most text mining tasks use Information Retrieval (IR) methods to pre-process text documents.
- These methods are quite different from traditional data pre-processing methods used for relational tables.
- Web search also has its root in IR.

## Information Retrieval (IR)

- Conceptually, IR is the study of finding needed information. I.e., IR helps users find information that matches their information needs.
  - Users' information needs are expressed as queries
- Information Retrieval (IR): retrieving desired information from documents.
- Historically, IR is about document retrieval, emphasizing (considering) document as the basic unit.
- Technically, IR studies the acquisition, organization, storage, retrieval, and distribution of information.




***The information retrieval process***

## Some information needs

- Searching for a topic
  - “climate change”
- Individual search
  - “The Ph.D thesis of Radwan Jalam”
- Searching for facts or events
  - “The Chancellor of the University of M’sila in 2002”
  - “The exact date of ISIA’14 University of M’sila”

## There exist many types of retrieval :

- Text retrieval
  - Match a query against free-text documents
  - Retrieve relevant documents
  - Retrieve relevant sentences
- Music retrieval
  - How to match a query to music?
- Image retrieval
  - Retrieve images from a large database of digital images

## Different kinds of documents

- Strictly structured (relational database)
  - E.g. library database, …
- Semi-structured
  - E.g. reference database: meta data for the publications and their abstracts.
  - XML documents
- Unstructured text documents
  - Word Documents, works documents, …

## Information needs vs. different kinds of documents

- Queries can be exact or approximate
  - – Exact query: relevant documents can be described with some features in an unambiguous (clear) way
  - – Approximate query: relevant documents cannot be described with some features or in an unambiguous way
- Exact database query

– “Students that major in computer science and started their studies in 2001”

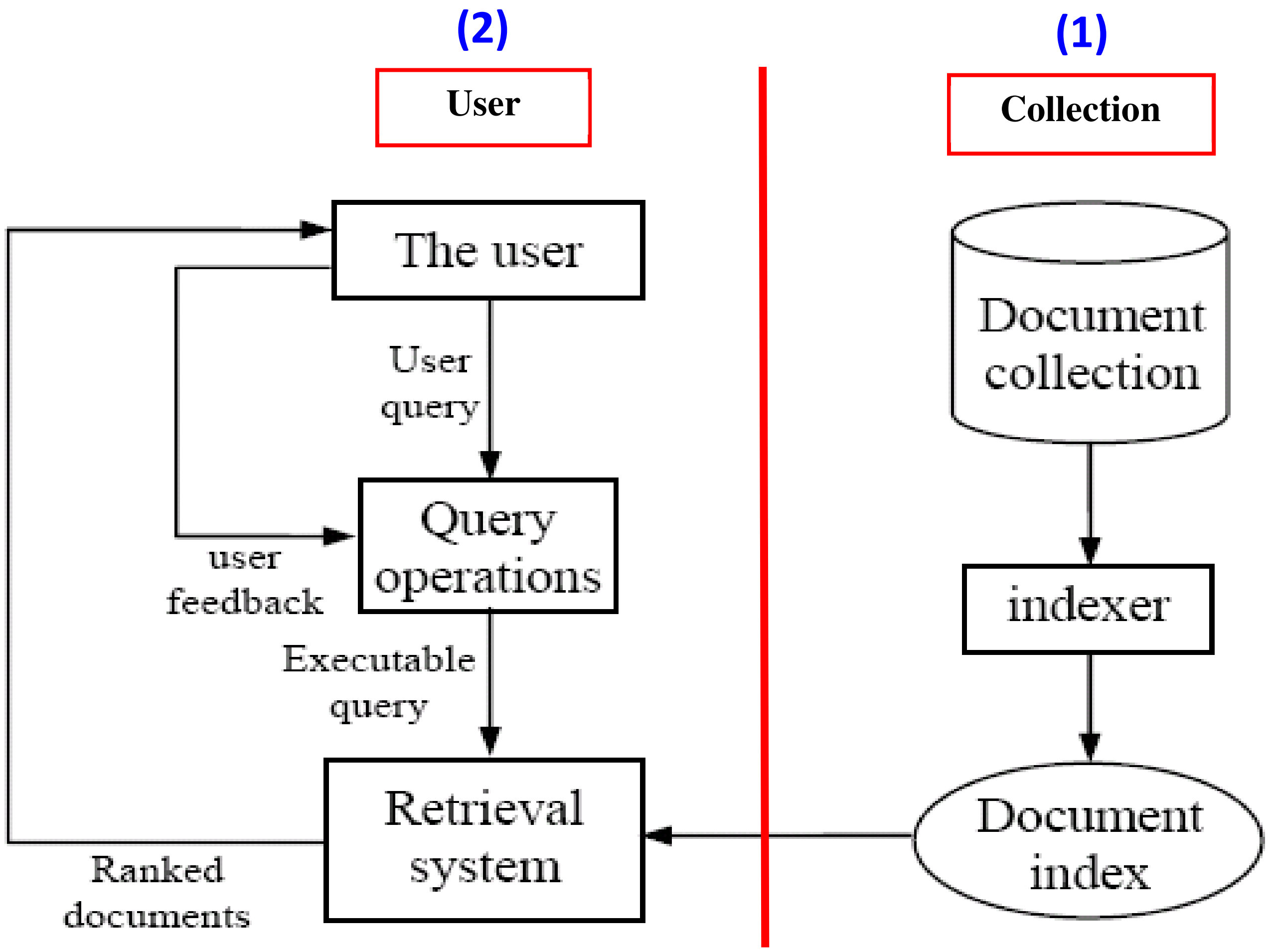
  - ➢ Attributes: first year, discipline
  - – The answer is always correct (unless the database contains errors)
- A database query can also be approximate
  - – The system could return students that started their studies in 2000-2002 (e.g. if there were no students starting in 2001)
- Queries on full text are usually approximate: (e.g. ”climate change”)
  - – It is hard to know which terms have been used in different documents that discuss this topic: “climate change global warming weather carbon emission, … “
  - – Many other topics may have been described with the same terms
    - ➢ The result is often incomplete
    - ➢ The result may contain irrelevant documents
- The result of a query may be direct or indirect
  - – Direct: the answer is found in the result

– Indirect: the result contains pointers to the sources of the information that was searched for, e.g., literature references to documents, addresses of companies, Website Links, etc.

- Queries on semi-structured documents combine exact and approximate queries

  – ”books written by John Irving containing a character named Jack”

## IR architecture



## Representations of queries and documents

both queries and documents must be represented in a more suitable ways:

- By a set of terms (term = a unit of semantic expression, e.g. word, phrase, stem (of a word))
- A document can also be represented
  – Automatically based on terms that have been selected from the document on statistical grounds.
  – Automatically based on terms that have been selected from the document on linguistic grounds.
  – With terms selected by a human expert.
- A document set can be very large
- There can be a very large number of terms (~ 10 000 – 100 000)

## IR queries

- Keyword queries
  - A set of index terms (key words)
- Boolean queries
  - An expression, where index terms have been combined with boolean operators (AND, OR, NOT)
    – "John and Irving"
    – "(text or image) and retrieval"
- Phrase queries: using a complete sentence between " ".
- Proximity queries
  - Terms combined with proximity (nearness) operators
    – "John near Irving"

- Full document queries.
  - The name of a retrieved document can act as a query
    - The system looks for documents that take this name.
- Natural language questions
  - Sentences in natural language
    - E.g. a question-answering system accepts questions as Input such as: “Who was the Chancellor of the University in 2002?”

## Concepts related to IR systems

- An IR model governs how a document and a query are represented and how the relevance of a document to a user query is defined.
- **Relevance**
  - True relevance: A relevant document meets user’s information need.
  - Relevance score: A numeric score assigned to a search result, representing how well the result “matchs” the query.
- **Similarity:** measure of how close a query is to a document.
- **Similarity Measures:**
  - Dice, Jaccard, Cosine, Overlap
  - The similarity is being evaluated between two vectors
- Documents which are “close enough” to a query are retrieved

## Main IR models

- Boolean model
- Vector space model
- Statistical language model
- Etc.

### 1. Boolean model

- Each document or query is treated as a **"bag" of words** or **terms**. Word sequence is not considered.
- Given a collection of documents $D$, let $V = \{t_1, t_2, ..., t_{|V|}\}$ be the set of distinctive (words/terms) in the collection. $V$ is called the vocabulary.
- A weight $w_{ij} > 0$ is associated with each term $t_i$ of a document $\mathbf{d}_j \in D$. For a term that does not appear in document $\mathbf{d}_j$, $w_{ij} = 0$.

  **===>** $\mathbf{d}_j = (w_{1j}, w_{2j}, ..., w_{|V|j})$.

- Query terms are combined logically in a query using the Boolean operators **AND**, **OR**, and **NOT.**
  - E.g., Q: "((*data* AND *mining*) AND (NOT *text*))"
- **Retrieval**
  - Given a Boolean query, the system retrieves every document that makes the query logically true.

- The retrieval results are usually quite poor because term frequency is not considered.

## 2. Vector space model

- Documents are also treated as a “bag” of words or terms.
- Each document is represented as a vector.
- However, the term weights are no longer 0 or 1. Each term weight is computed based on some variations of TF or TF-IDF scheme.
- **Term Frequency (TF) Scheme:** The weight of a term $t_i$ in document $\mathbf{d}_j$ is the number of times that $t_i$ appears in $\mathbf{d}_j$, denoted by $f_{ij}$. A normalization may also be applied.

**NB:**

- **N**ot all words in a document are equally important.
- Terms occurring very often in the collection (in many documents in the same time) are not relevant for distinguishing among the documents.
- A relevance measure cannot only take term frequency into account.

  ⇨ ***Reducing the weight of a term*** using a factor growing with the collection frequency.

## Collection frequency versus document frequency

| Term t | $Cf_t$ | $Df_t$ |
|---|---|---|
| Try | 10422 | 8760 |
| Insurance | 10440 | 3997 |

## Normalized TF

- We can apply a normalization on TF.

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{|V|j}\}}$$

## Inverse document frequency of a term t (IDF)

- A measure of the general importance of the term.
- It is the logarithm of the number of all documents divided by the number of documents containing the term.

$$idf_i = \log \frac{N}{df_i}$$

N: the total number of docs (collection size).

$df_i$: the number of docs that $t_i$ appears in.

**Remark:**

Rare terms have high idf, contrary to frequent terms.

**Example** (From: Reuters collection; ***Manning et al.***):

| **Term t** | **$df_t$** | **$Idf_t$** | **Size of collection** |
|---|---|---|---|
| Try<br>Insurance | 18165<br>19241 | 1.65<br>1.62 | 806791 docs |

**TF-IDF term weighting scheme**

- Evaluate how important a word is to a document in a collection
- Many different ways to calculate the weight
- High value of Tf-Idf weight is reached by a high tf and a low Idf, filter out common terms
- The weight of a term is computed using both tf and idf

$$w_{ij} = tf_{ij} \times idf_i .$$

- Where $w_{ij}$ is:
  1. high when *t* occurs many times in a small set of documents
  2. low when *t* occurs few times in a document, or when it occurs in many documents

3. very low when *t* occurs in almost every document

- Score of a document with respect to a query:

$$score(q, d) = \sum_{t \in q} w(t, d)$$

## Vector normalization and similarity

- Euclidian normalization (vector length normalization):

$$v(\vec{d}) = \frac{V(\vec{d})}{\|V(\vec{d})\|}$$

Where:

$$\|V(\vec{d})\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

- Similarity given by the cosine measure between normalized vectors:

$$Sim(d_i, d_j) = Cos(d_i, d_j)$$

- Cosine similarity (the cosine of the angle between the two vectors)

$$cosine(\mathbf{d}_j, \mathbf{q}) = \frac{\langle \mathbf{d}_j \bullet \mathbf{q} \rangle}{\| \mathbf{d}_j \| \times \| \mathbf{q} \|} = \frac{\sum_{i=1}^{|V|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}}$$

■ Cosine is also commonly used in text clustering

## Example

| Dictionary | V ($d_1$) | V ($d_2$) | V ($d_3$) |
|---|---|---|---|
| player | 0.996 | 0.993 | 0.847 |
| Games | 0.087 | 0.120 | 0.466 |
| Vehicle | 0.017 | 0 | 0.254 |

Sim($d_1$, $d_2$) = Cos($d_1$, $d_2$) $\cong$ 0.999

Sim($d_1$, $d_3$) $\cong$ 0.888

Sim($d_2$, $d_3$) $\cong$ 0.897

## Retrieval in vector space model

- Query **q** is represented in the same way or slightly differently
- Relevance of $\mathbf{d}_i$ to **q**: Compare the similarity of query **q** and document $\mathbf{d}_i$.
- Cosine similarity is often used.
- Compute the similarity ***(q, d$_i$)***

- Selection of the top *K* scores.
- The use of $tf - idf_{t,d}$ measure as a weight.

## An Example

- A document space is defined by three terms:
  - ✓ hardware, software, users.
  - ✓ the vocabulary
- A set of documents are defined as:

  A1=(1, 0, 0), A2=(0, 1, 0), A3=(0, 0, 1)

  A4=(1, 1, 0), A5=(1, 0, 1), A6=(0, 1, 1)

  A7=(1, 1, 1) A8=(1, 0, 1). A9=(0, 1, 1)

- If the Query is "***hardware and software***"

  ***What documents should be retrieved?***

- **In Boolean query matching:**
  - Document A4, A7 will be retrieved ("AND")
  - Retrieved: A1, A2, A4, A5, A6, A7, A8, A9 ("OR")
- **In Vector Space model - similarity matching (cosine):**
  - q = (1, 1, 0)
  - S(q, A1)=0.71, S(q, A2)=0.71, S(q, A3)=0
  - S(q, A4)=1, S(q, A5)=0.5, S(q, A6)=0.5
  - S(q, A7)=0.82, S(q, A8)=0.5, S(q, A9)=0.5

- Document retrieved set (with ranking) =

  {A4, A7, A1, A2, A5, A6, A8, A9}

## Other similarity measures

- Many similarity measures are used in data mining to mine text/web data and also used in IR.

$$\textbf{Dice:}\quad sim(t_i, t_j) = \frac{2\sum_{h=1}^{k} t_{ih}t_{jh}}{\sum_{h=1}^{k} t_{ih}^2 + \sum_{h=1}^{k} t_{jh}^2}$$

$$\textbf{Jaccard:}\quad sim(t_i, t_j) = \frac{\sum_{h=1}^{k} t_{ih}t_{jh}}{\sum_{h=1}^{k} t_{ih}^2 + \sum_{h=1}^{k} t_{jh}^2 - \sum_{h=1}^{k} t_{ih}t_{jh}}$$

$$\textbf{Cosine:}\quad sim(t_i, t_j) = \frac{\sum_{h=1}^{k} t_{ih}t_{jh}}{\sqrt{\sum_{h=1}^{k} t_{ih}^2 \sum_{h=1}^{k} t_{jh}^2}}$$

$$\textbf{Overlap:}\quad sim(t_i, t_j) = \frac{\sum_{h=1}^{k} t_{ih}t_{jh}}{min(\sum_{h=1}^{k} t_{ih}^2, \sum_{h=1}^{k} t_{jh}^2)}$$

## Okapi relevance method

- Another way to assess the degree of relevance is to directly compute a relevance score for each document to the query.
- The Okapi method and its variations are popular techniques in this setting.

The Okapi relevance score of a document $d_j$ for a query $q$ is:

$$okapi(d_j, q) = \sum_{t_i \in q, d_j} \ln \frac{N - df_i + 0.5}{df_i + 0.5} \times \frac{(k_1 + 1) f_{ij}}{k_1 (1 - b + b \frac{dl_j}{avdl}) + f_{ij}} \times \frac{(k_2 + 1) f_{iq}}{k_2 + f_{iq}},$$

where $k_1$ (between 1.0-2.0), $b$ (usually 0.75) and $k_2$ (between 1-1000)

## Scoring Optimization

### Relevance feedback

- Relevance feedback is one of the techniques for improving retrieval effectiveness. It is done in many steps:
  - ❑ The user first identifies some relevant ($D_r$) and irrelevant documents ($D_{ir}$) in the initial list of retrieved documents.
  - ❑ The system expands the query **q** by extracting some additional terms from the sample relevant and irrelevant documents to produce $\mathbf{q}_e$
  - ❑ Perform a second round of retrieval.
- Using Rocchio method ($\alpha$, $\beta$ and $\gamma$ are parameters)

$$\mathbf{q}_e = \alpha \mathbf{q} + \frac{\beta}{|D_r|} \sum_{\mathbf{d}_r \in D_r} \mathbf{d}_r - \frac{\gamma}{|D_{ir}|} \sum_{\mathbf{d}_{ir} \in D_{ir}} \mathbf{d}_{ir}$$

## Rocchio text classifier

- In fact, a variation of the Rocchio method above, called the **Rocchio classification** method, can be used to improve retrieval effectiveness too
  - ❑ So are other machine learning methods. Why?
- Rocchio classifier is constructed by producing a prototype vector $\mathbf{c}_i$ for each class $i$ (*relevant* or *irrelevant* in this case):

$$\mathbf{c}_i = \frac{\alpha}{|D_i|} \sum_{\mathbf{d} \in D_i} \frac{\mathbf{d}}{\|\mathbf{d}\|} - \frac{\beta}{|D - D_i|} \sum_{\mathbf{d} \in D - D_i} \frac{\mathbf{d}}{\|\mathbf{d}\|}$$

- In classification, the cosine similarity measure is used.

# Some important tasks to perform before retrieval

## Document pre-processing

- Word (term) extraction: easy
- Stopwords removal
- Stemming
- Frequency counts and computing TF-IDF term weights.

## Stopwords removal

- Many of the most frequently used words in English are useless in IR and text mining – these words are called *stop words*.
  - ❑ The, of, and, to, ….
  - ❑ Typically, about 400 to 500 such words
  - ❑ For an application, an additional domain specific stopwords list may be constructed
- Why do we need to remove stopwords?
  - ❑ Reduce indexing (or data) file size
    - stopwords accounts 20-30% of total word counts.
  - ❑ Improve efficiency and effectiveness
    - stopwords are not useful for searching or text mining
    - they may also confuse the retrieval system.

## Stemming

- Techniques used to find out the root/stem of a word. E.g.,
  - user — engineering
  - users — engineered
  - used — engineer
  - using
- Stem: use — engineer

**Usefulness:**

- Improving effectiveness of IR and text mining
  - Matching similar words
  - Mainly improve recall
- Reducing indexing size
  - Combing words with same roots may reduce indexing size as much as 40-50%.

**Basic stemming methods**

Using a set of rules. E.g.,

- ***Remove ending***
  - if a word ends with a consonant other than s, followed by an s, then delete s.

- if a word ends in es, drop the s.
- if a word ends in ing, delete the ing unless the remaining word consists only of one letter or of th.
- If a word ends with ed, preceded by a consonant, delete the ed unless this leaves only a single letter.
- ……

- ***Transform words***
  - if a word ends with “ies” but not “eies” or “aies” then “ies --> y.”

## Frequency counts + TF-IDF

- Counts the number of times a word occurred in a document.
  - Using occurrence frequencies to indicate relative importance of a word in a document.
    - if a word appears often in a document, the document likely “deals with” subjects related to the word.
- Counts the number of documents in the collection that contains each word
- TF-IDF can be computed.

## Question Answering Vs Information Retrieval

- Question in QA vs. Query in IR
- Exact answers vs. relevant documents for results
- A typical QA system
  - – Question analysis
  - – Search engine (a revised IR system)
  - – Answer extraction

## Question Answering System

- Require more complex natural language processing techniques
  - ***– Named entity Extraction, Parser, Part-of-speech tagger***
- A wide range of question:
  - ***– Factoid: (person, location, date, organization, money amount, etc.)***
  - ***– List, why, how, definition***
- Closed-domain QA & Open-domain QA
  - ***– Domain specific knowledge, accept a limited type of questions***
  - ***– General knowledge, user can basically ask any type of question***

## Example of questions in QA system

- Who is the president of Turkey?
- How many members are in the UNO council?
- When was El-Amir Abdelkader born?
- Name 22 cities that have a subway system?

## Web Search Engine

- An important application of IR
- Search in the large amount of data on the web
- Pages with heterogeneous data and extensive hyperlinks
  - Multi-language
  - Various sources
  - Different formats
  - …

## Web Search Engines - problems

- Abundance
  - Too much data, a query only retrieves a small subset of it
- Limited coverage
  - Search a result from a subset of the web.
  - Only periodically update the index

■Limited query

– Key-word based searching, works for short queries

■ Limited customization

– Query results determined by the query itself.

– IR systems consider the background and knowledge of the users as well.

## Cross-Lingual Retrieval

■ Accepting query in one language (English), retrieving documents in another language (French).

■ Typical approach is to translate query and then use monolingual search engines

■ Language resources

– Bilingual dictionary, parallel collection, MT systems.

■ Major issue: Translation ambiguity

– Multiple translations for each word

– Translation probabilities required for some approaches

## Web Search as a huge IR system

- A Web crawler (robot) crawls the Web to collect all the pages.
- Servers establish a huge inverted indexing database and other indexing databases
- At query (search) time, search engines conduct different types of vector query matching.

## Web Search Systems

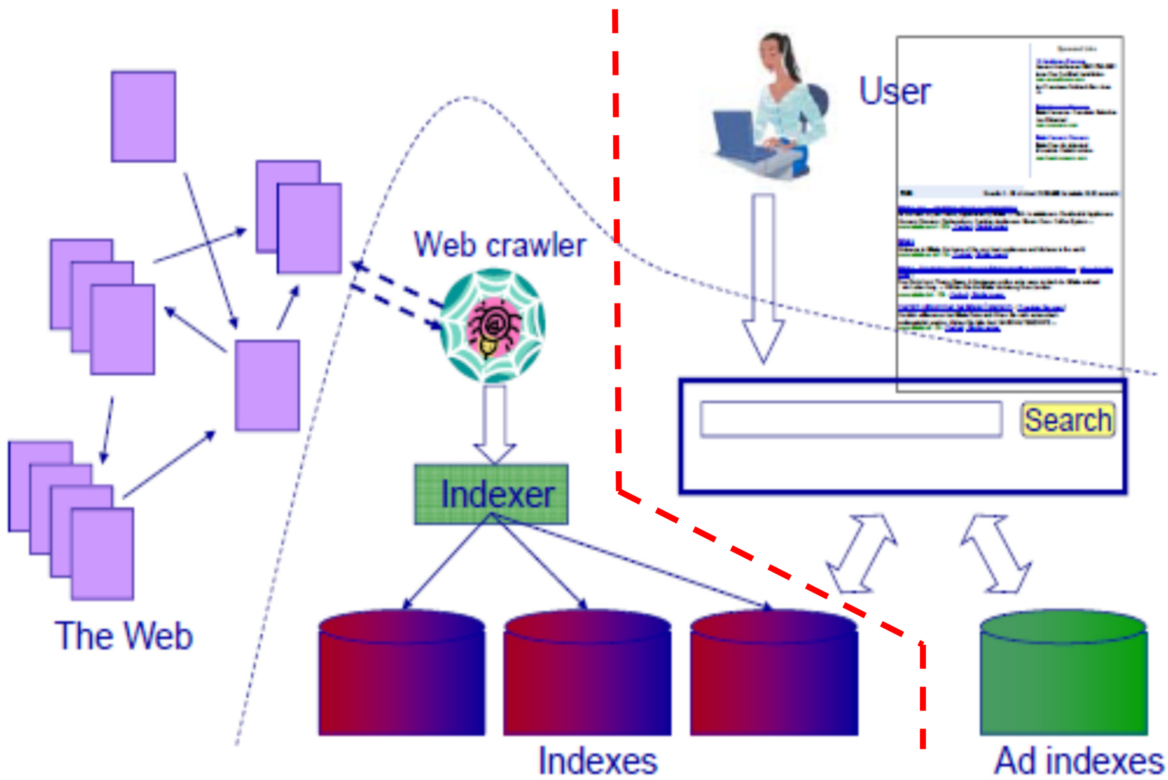



**Figure 2.6. Basics of Web Search**

## Differences between search engines

The real differences among different search engines are:

- ❑ Their index weighting schemes
  - Including location of terms, e.g., title, body, emphasized words, etc.
- ❑ Their query processing methods (e.g., query classification, expansion, etc)
- ❑ Their ranking algorithms
  - Few of these are published by any of the search engine companies. They are tightly guarded secrets.

## Web Search

### Background and history

- Complexity of web search comes from:
  - ✓ Its scale (about 20 billion pages currently)
  - ✓ Its lack of coordination (decentralized content publishing)
  - ✓ The heterogeneity of its contributors (motives, backgrounds)

- Success of WWW comes from:
  - ✓ Easy-to-learn edition language (HTML)
  - ✓ Robust browsers (unknown code is ignored)

## Early web search engines and web collections

■Two families of engines:

1.Full-text index-based search engines (altavista, excite, infoseek)

2. Taxonomy-based search engines (yahoo)

■ Early collections:

- Tens of millions of pages (larger than any prior collection)
  - ✓ Indexing and fast querying performed successfully, without the expected quality of retrieval
  - ✓ New techniques were needed to rank the retrieved pages and deal with the spams.

## Indexing the web

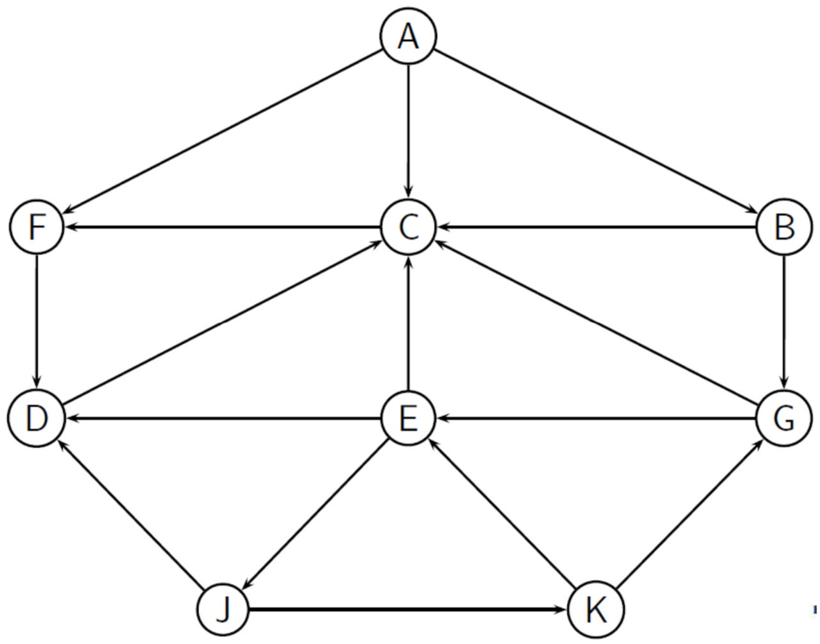
■ Questions arising when one wants to index the web:

- Which pages can one trust (have confidence)?
- How a search engine can assign a measure of trust (confidence) to a webpage?
- How to deal with the expansion of the collection?
- How to deal with redundancy

■ By the end of 1995, Altavista had crawled 30 million static webpages, after that the size of the index was multiplied by 2 every few months)

## The web as a graph

- The web can be represented as a graph:
    - ✓ webpage ≡ node
    - ✓ hyperlink ≡ directed edge
    - ✓ # in-links ≡ in-degree of a node
    - ✓ # out-links ≡ out-degree of a node




The web has a Bowtie shape (une forme de papillon) where webpages belong to one of these categories:

- **Strongly Connected Component (SCC)**

  - Set of pages that can be reached by one another along directed links.

  - About 30% of the Web (normal pages)

■ **IN Group**

- Set of pages that have a path to SCC but not from it.
- About 20% (maybe new pages or boring ones)

■ **OUT Group**

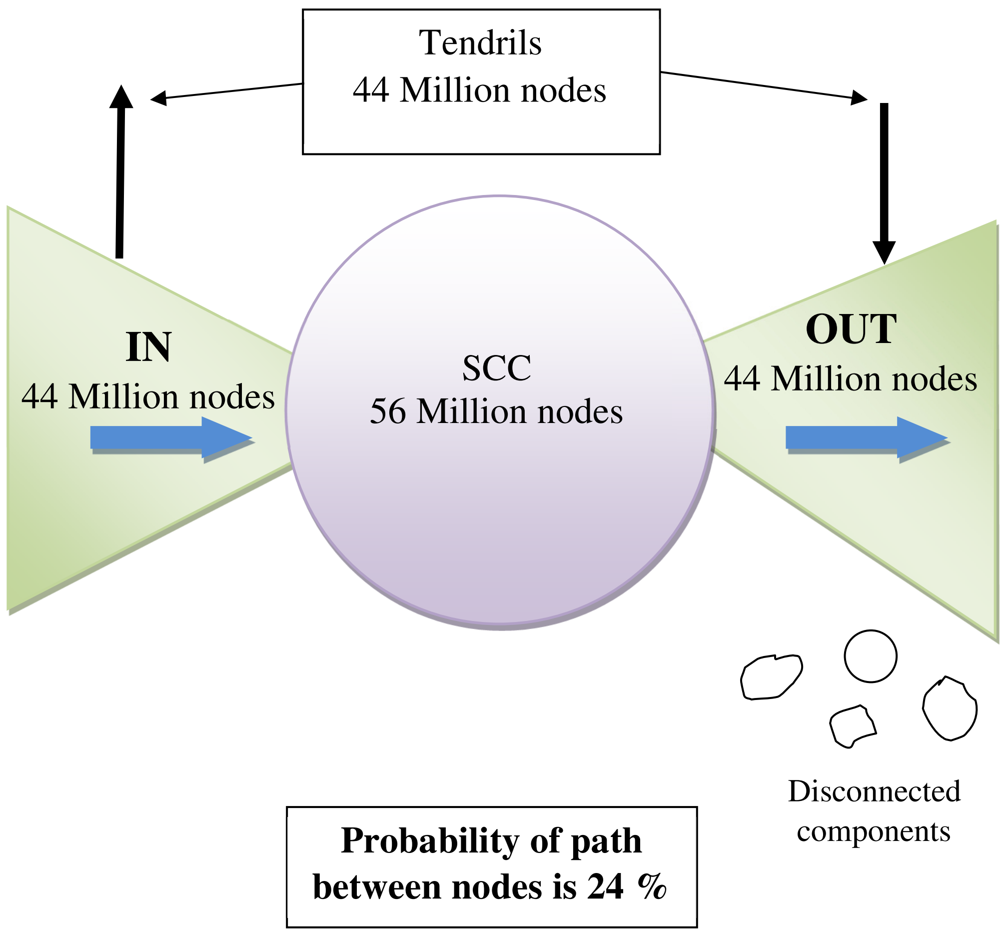
- Set of pages that can be reached from SCC but cannot reach it.
- About 20% (may be company pages that don't link)

■ **Tendrils**

- Cannot reach SCC and cannot be reached by it (about 20%)

■ **Unconnected**

- About 10%



## Spam in IR

1. First generation of spam: building documents with specific high-frequency terms, in order to appear first in the retrieval for some queries
2. A doorway document is used to get highly ranked, but when accessed by a browser, it redirects the user to a spam

## Different types of web search users

■ Improving retrieval results needs to better understand how the search engine is used (kind of users):

- Users do not know (or care) about the heterogeneity of content
- Users do not know (or care) about the querying syntax
- Users use on average between 2 and 3 keywords

■ Catching a bigger audience needs to better understand how the search engine is used.

- The google example:

  (1) focus on relevance and precision (rather than recall)

  (2) lightweight user experience.