Information Retrieval

finding relevant information
in a store of information.

IR does not mean finding
any information which we happen to come across,
or information we are fortunate enough
to discover by chance without having in mind anything particular.

IR means that we already have
a need for information that we are able to formulate,
and then relevant items are found in a store (collection) of items.

History of Information Retrieval

❖In the late 1940s,

the United States of America military carried out an indexing of wartime scientific research documents captured from Germans.

❖In the 1950s,

the Soviet Union sent up the first artificial Earth satellite.

“science gap”

This motivated American funding of research in mechanized literature search.

❖At the end of 1980s,

the World Wide Web was proposed.

The Web is a worldwide network of electronic documents stored in computers belonging to the Internet.

‘Definition’ of Information Retrieval (IR)

IR is concerned with

the organisation,

storage,

retrieval and

evaluation

of information relevant to a user’s information need.

The user

has an information need:

- (e.g., articles published on a certain subject
- travel agencies with last minute offers,
- etc.)

The information need is expressed in the form of a query, i.e., in a form which is required by a computer program.

The program then retrieves information (journal articles, Web pages, etc.) in response to the query.

The term *IR* may be formulated formally:

$$\boldsymbol{IR = (U, IN, Q, O) \rightarrow R,}$$

where

- *U* = user,
- *IN* = information need,
- *Q* = query,
- *O* = collection of objects to be searched,
- *R* = collection of retrieved objects in response to *Q*.

- The *IN* is more than its expression in a query *Q*.
- *IN* comprises the query *Q* plus additional information about the user *U*.
- The additional information is specific to the user
- The additional information is obvious for the user but not for the computerized retrieval system.

⇓

Thus the additional information is an implicit (i.e., not expressed in *Q*) information *I* specific to the user *U*, and we may write ***IN* = (*Q*, *I*).**

A more strict re-formulation of the meaning of *IR:*

IR is being concerned with finding

- ❖a relevance relationship $\Re$
- ❖between objects *O* and information need *IN*;

formally:

$$\boldsymbol{IR = \Re(O, IN) = \Re(O, (Q, I)).}$$

To find such a relationship $\Re$:

- ❖it should be made possible to take into account the implicit information *I* as well,
- ❖and ideally the information which can be inferred from *I* to obtain as complete a picture of user *U* as possible.

Finding an appropriate relationship $\Re$ would mean to obtain (derive, infer) those objects *O:*

- ❖which match the query *Q* and
- ❖satisfy the implicit information *I*.

With these, the notion of *IR* re-writes formally as follows:

$$IR = \Re(O, (Q, \langle I, \vdash\!\!\!\rangle)),$$

where $\langle I, \vdash\!\!\!\rangle$ means I together with information inferred (e.g., in some formal language or logic) from I. Relationship $\Re$ is established with some (un)certainty m, and thus we may write that:

$$IR = m[\Re(O, (Q, \langle I, \vdash\!\!\!\rangle))].$$

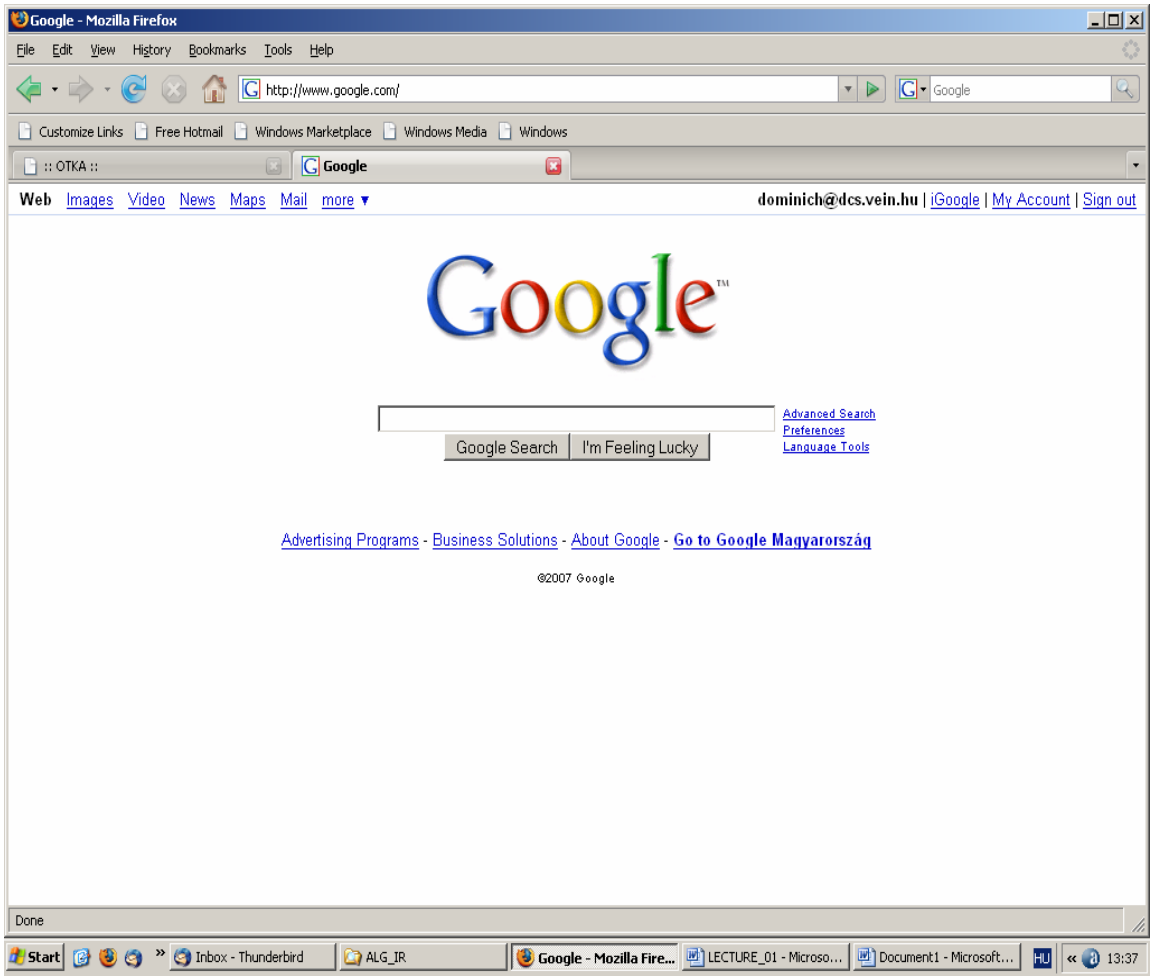
$$\Downarrow$$

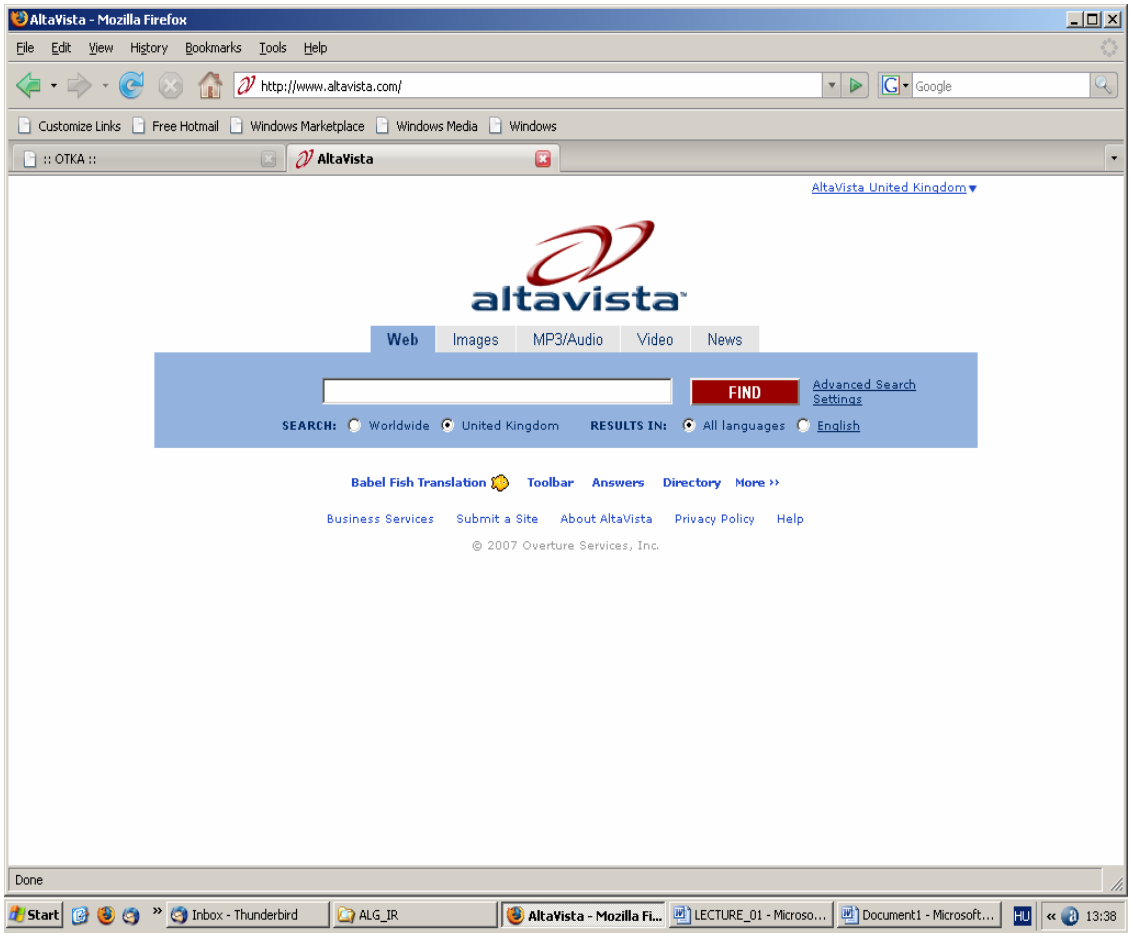
Information Retrieval is a kind of measurement

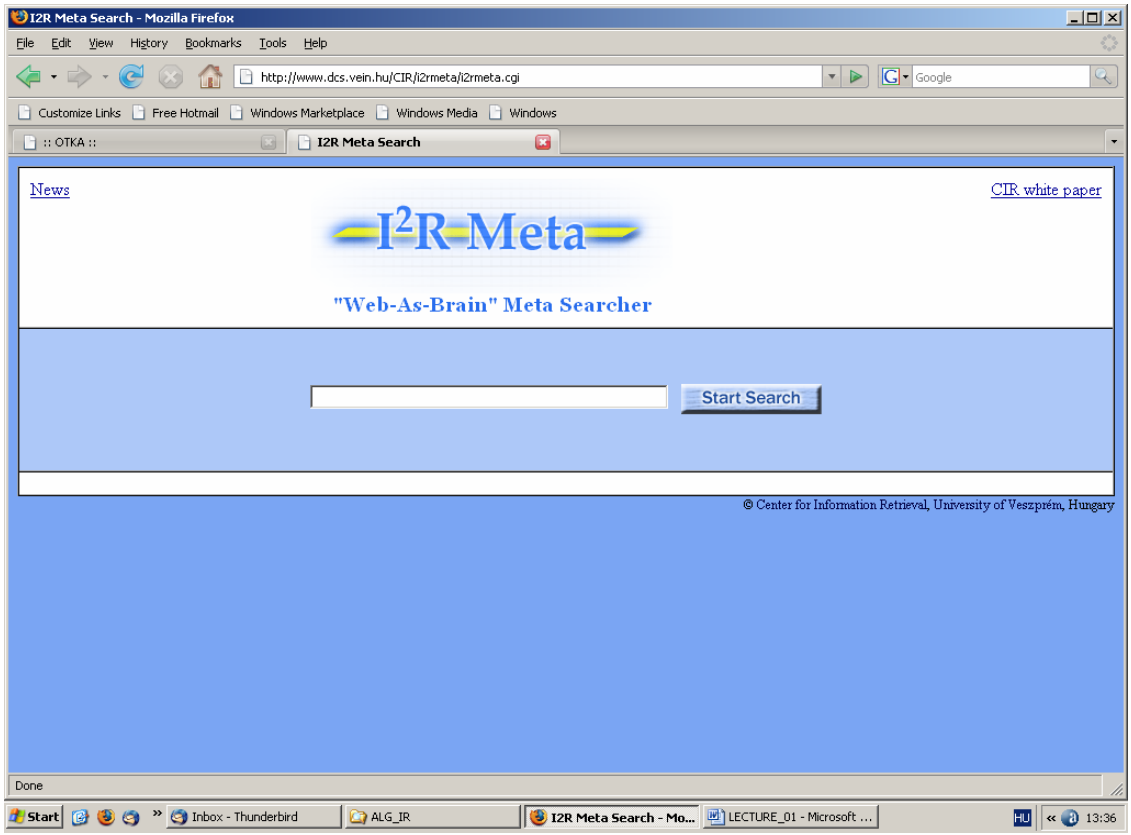
measuring the relevance of an item stored in computer memory to an user's information request (and then returning the items sorted descendingly on their measure of relevance).

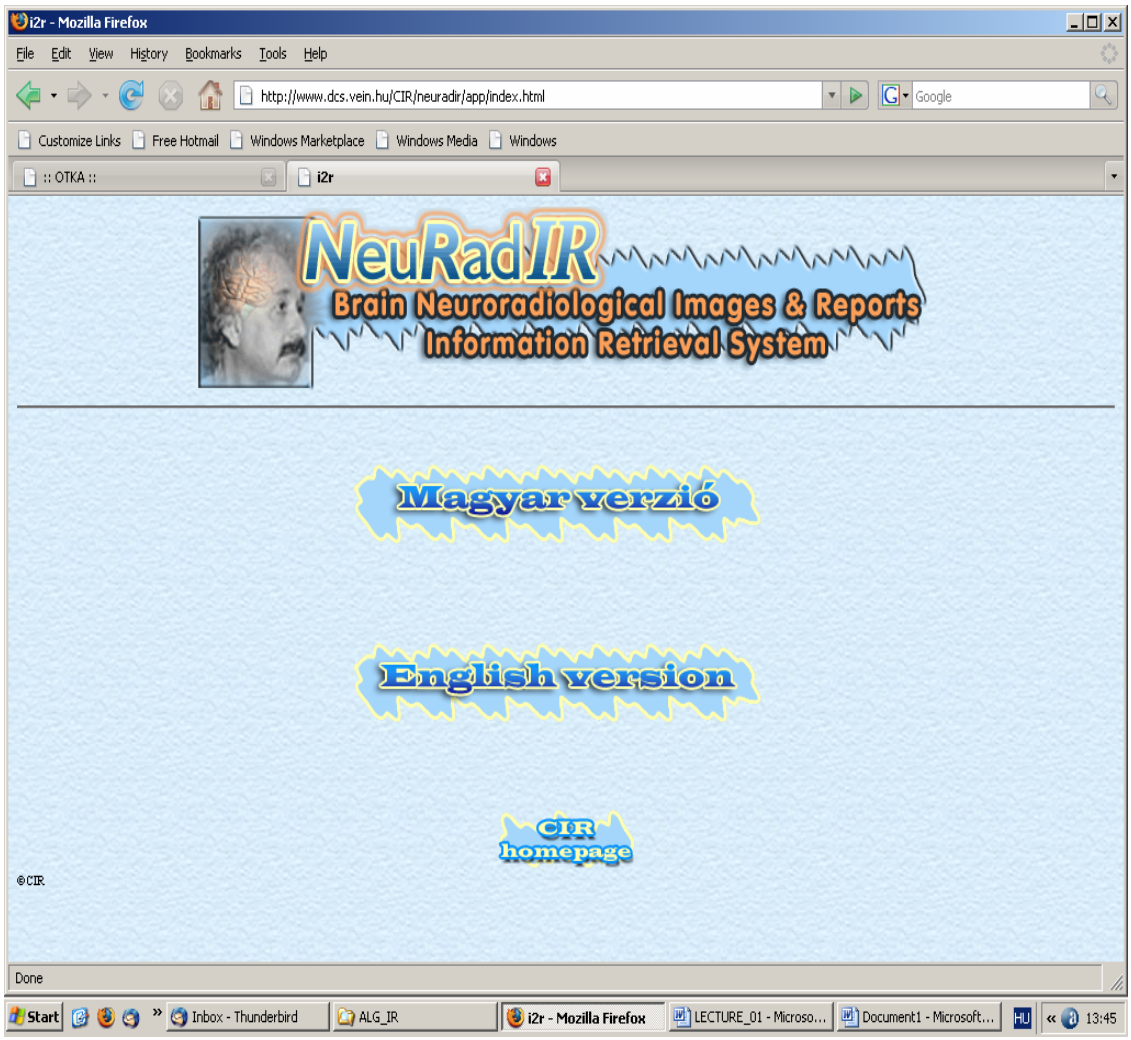
All *IR* frameworks, methods, and algorithms aim at as good a measurement as possible.

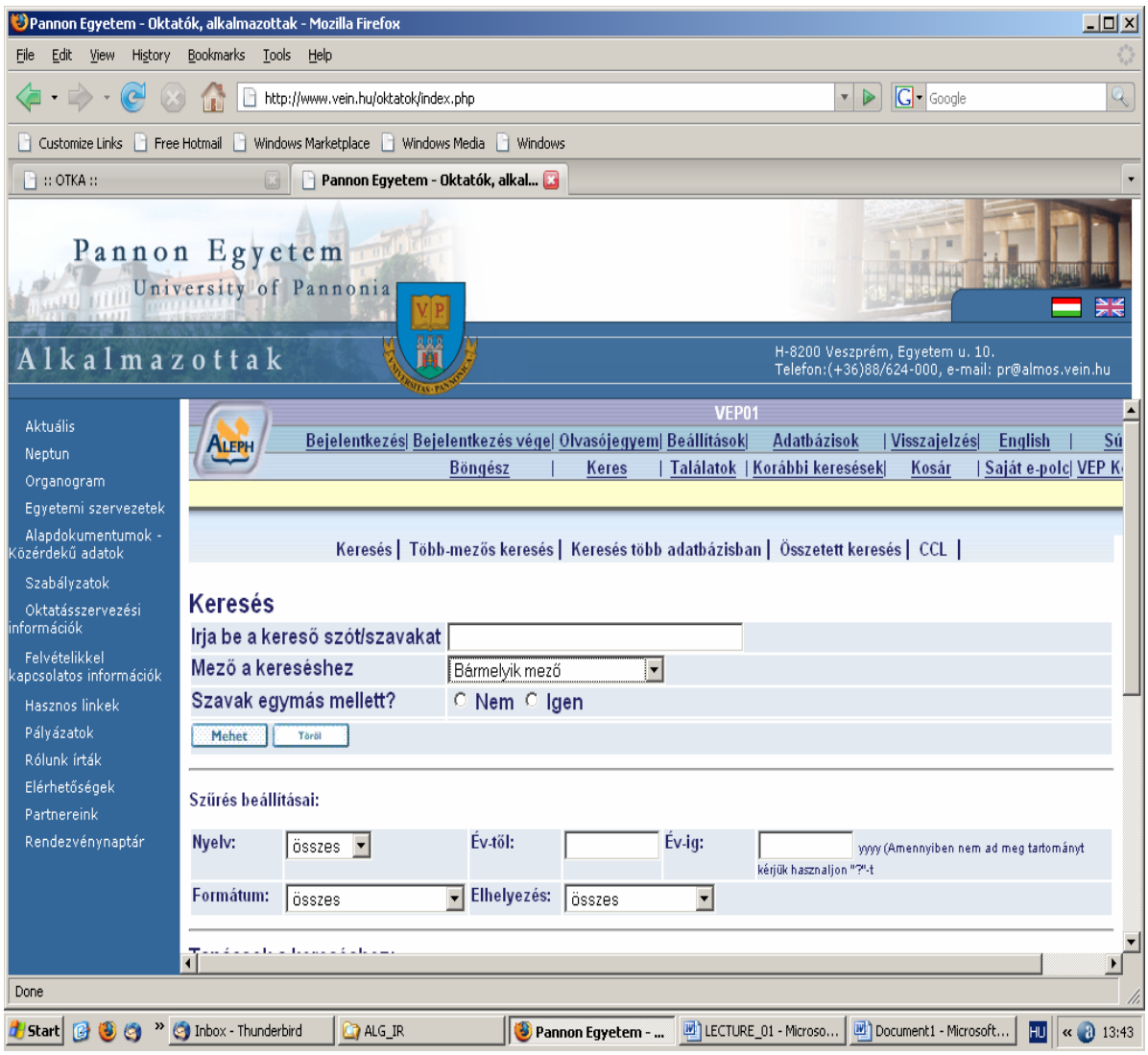
INFORMATION RETRIEVAL APPLICATIONS




AltaVista - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.altavista.com/
Customize Links Free Hotmail Windows Marketplace Windows Media Windows
:: OTKA ::
AltaVista
AltaVista United Kingdom
altavista
Web Images MP3/Audio Video News
FIND
Advanced Search
Settings
SEARCH: Worldwide United Kingdom
RESULTS IN: All languages English
Babel Fish Translation Toolbar Answers Directory More ››
Business Services Submit a Site About AltaVista Privacy Policy Help
© 2007 Overture Services, Inc.
Done
Start Inbox - Thunderbird ALG_IR AltaVista - Mozilla Fi... LECTURE_01 - Microso... Document1 - Microsoft... HU 13:38


I2R Meta Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.dcs.vein.hu/CIR/i2rmeta/i2rmeta.cgi
Customize Links Free Hotmail Windows Marketplace Windows Media Windows
:: OTKA ::
I2R Meta Search
News
CIR white paper
I2R Meta
"Web-As-Brain" Meta Searcher
Start Search
© Center for Information Retrieval, University of Veszprém, Hungary
Done
Start Inbox - Thunderbird ALG_IR I2R Meta Search - Mo... LECTURE_01 - Microsoft ... HU 13:36


i2r - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.dcs.vein.hu/CIR/neuradir/app/index.html
Google
Customize Links Free Hotmail Windows Marketplace Windows Media Windows
:: OTKA ::
i2r
NeuRadIR
Brain Neuroradiological Images & Reports
Information Retrieval System
Magyar verzió
English version
CIR homepage
©CIR
Done
Start
Inbox - Thunderbird
ALG_IR
i2r - Mozilla Firefox
LECTURE_01 - Microso...
Document1 - Microsoft...
HU
13:45


Pannon Egyetem - Oktatók, alkalmazottak - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.vein.hu/oktatok/index.php
Customize Links Free Hotmail Windows Marketplace Windows Media Windows
:: OTKA ::
Pannon Egyetem - Oktatók, alkal...
Pannon Egyetem
University of Pannonia
Alkalmazottak
H-8200 Veszprém, Egyetem u. 10.
Telefon:(+36)88/624-000, e-mail: pr@almos.vein.hu
Aktuális
Neptun
Organogram
Egyetemi szervezetek
Alapdokumentumok - Közérdekű adatok
Szabályzatok
Oktatásszervezési információk
Felvételikkel kapcsolatos információk
Hasznos linkek
Pályázatok
Rólunk írták
Elérhetőségek
Partnereink
Rendezvénynaptár
VEP01
Bejelentkezés | Bejelentkezés vége | Olvasójegyem | Beállítások | Adatbázisok | Visszajelzés | English
Böngész | Keres | Találatok | Korábbi keresések | Kosár | Saját e-polc | VEP K
Keresés | Több-mezős keresés | Keresés több adatbázisban | Összetett keresés | CCL
Keresés
Irja be a kereső szót/szavakat
Mező a kereséshez
Bármelyik mező
Szavak egymás mellett?
Nem Igen
Mehet
Töröl
Szűrés beállításai:
Nyelv:
összes
Év-től:
Év-ig:
yyyy (Amennyiben nem ad meg tartományt kérjük használjon "?"-t
Formátum:
összes
Elhelyezés:
összes
Done
Start
Inbox - Thunderbird
ALG_IR
Pannon Egyetem - ...
LECTURE_01 - Microso...
Document1 - Microsoft...
HU
13:43

Documents

Let $E_1,\ldots,E_j,\ldots,E_m$ denote entities in general. E.g.,

- texts (books, journal articles, newspaper articles, papers, lecture notes, abstracts, titles, etc.),
- images (photographs, pictures, drawings, etc.),
- sounds (musical pieces, songs, speeches, etc.),
- multimedia (a collection of texts, images and sounds),
- a collection of Web pages,
- etc..

For retrieval purposes:

each entity E_j is described by a piece of text D_j.

Obviously, D_j may coincide with E_j itself

(for example, when E_j is itself a piece of text).

D_j is traditionally called a *document*.

Power Law

From a computational point of view:

The documents consist of words as automatically identifiable lexical units:

lexical unit = word =

string of characters preceded and followed by "space" (or some other character, e.g.: !, ., ?).

Thus, words can be recognised automatically (using a computer program).

The number f of occurrences of words in an English text (corpus) obeys a **Power Law**, i.e.,

$$f(r) = Cr^{-\alpha},$$

- where C is a corpus-dependent constant,
- and r is the rank of words.
- α is referred to as the *exponent* of the Power Law.

The Power Law $f(r) = Cr^{-1}$ is called Zipf Law.

For visualisation purposes, the Power Law is represented in a log-log plot, i.e., as a straight line obtained by taking the logarithm:

$$\log f(r) = \log C - \alpha \times \log r,$$

where

- $\log r$ is represented on the abscissa,
- $\log f(r)$ on the ordonata,
- $-\alpha$ is the slope of the line,
- and $\log C$ is the intercept of the line.

In practice, the following *regression method* can be applied to fit a Power Law to data:

Power Law Fitting Using Regression Method

1. Given a sequence of values $X = (x_1,...,x_i,...,x_n)$ on the horizontal axis, and another sequence of corresponding values $Y = (y_1,...,y_i,...,y_n)$ on the vertical axis (y_i corresponds to x_i).

2. If the correlation coefficient

$$r(X, Y) = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{\left(n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right)\left(\left(n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right)\right)}}$$

suggests a fairly strong correlation — it is close to +1 or −1 — between X and Y at a log scale, then a regression line can be drawn to exhibit a relationship between the data X and Y.

3. Using the

$$\text{slope} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

and the

$$\text{intercept} = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

of the regression line, the corresponding Power Law can be written.

Power Law Fitting by Least Squares Method

1. Given a sequence of values $X = (x_1,...,x_i,...,x_n)$ on the horizontal axis, and another sequence of corresponding values $Y = (y_1,...,y_i,...,y_n)$ on the vertical axis (y_i corresponds to x_i).

2. The parameters α and C should be so computed as to minimize the squared error

$$\sum_{i=1}^{n}\left(f(x_i)-y_i\right)^2=\sum_{i=1}^{n}\left(Cx_i^{-\alpha}-y_i\right)^2.$$

i.e., the partial derivatives with respect to C and α should vanish.

Example

Let us assume that the data we want to approximate by a Power Law is X and Y, $n = 150$.

Fragments of X and Y are shown below.

X =

	1
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

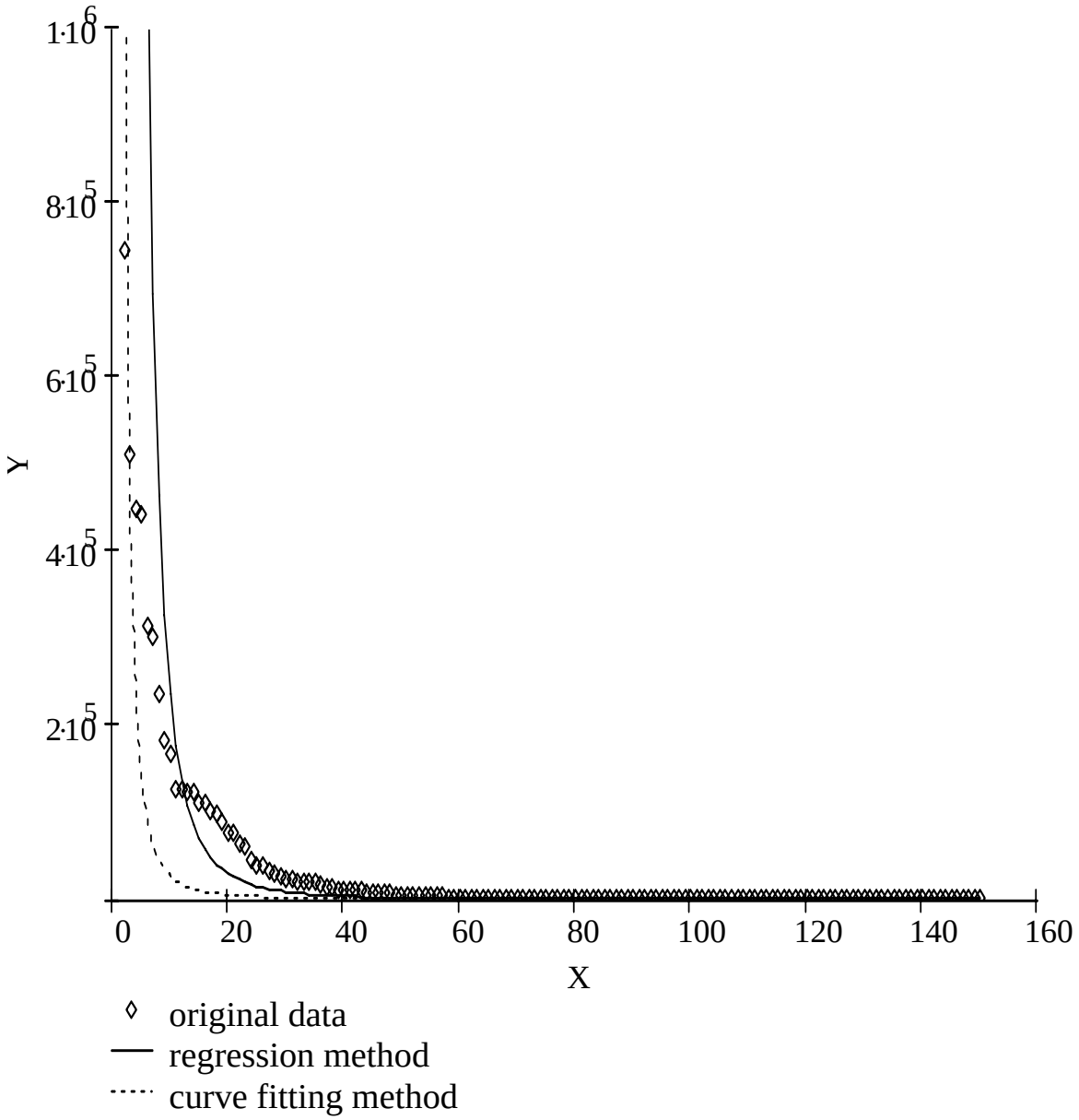
Y =

	1
1	$5.722975 \cdot 10^6$
2	744343.1
3	510729.7
4	449741.1
5	441213
6	313464.3
7	300948.4
8	235022.1
9	182827.1
10	167201.1

The correlation coefficient is $corr(X, Y) = -0.95$.

- Using the regression method, the following power law is obtained: $f(x) = 10^{8.38}x^{-3}$.
- Using the least squares method, the following power law is obtained: $f(x) = 5677733x^{-2.32}$.
- The approximation error is:
  - 2.8×10^8 in the regression method, and
  - 3.6×10^6 in the curve fitting method.

Thus, we should accept the Power Law obtained by the curve fitting method.


1·10^6
8·10^5
6·10^5
4·10^5
2·10^5
Y
0
20
40
60
80
100
120
140
160
X
original data
regression method
curve fitting method

Stoplist

In a document there are words:

- which occur many times, and
- there are words which occur once or just a few times.

One may disregard

- frequently (i.e., the frequency f exceeds some threshold value) occurring words on the ground that they are almost always insignificant, and
- infrequent words too (i.e., the frequency f is below some threshold value) on the ground that they are not much on the writer's mind (or else they would occur more frequently).

$$\Downarrow$$

stoplist

(list of frequent and infrequent words)

For the English language, a widely accepted and used stoplist is the so-called TIME stoplist:

TIME stoplist:

A
ABOUT
ABOVE
ACROSS
…
BACK
BAD
BE
…

A fragment for a Hungarian stoplist is as follows:

A
És
Az
Van
Is
Mely
Ez
Hogy
…

The construction of a stoplist can be automatised.

Other stoplists can also be used depending on, for example, the topic of the documents under focus.

Usually, one starts from a general stoplist, and enlarge/modify it according to topic, or depending on experimental results.

Stemming

Stemming:

words be transformed to their lexical roots.

Example:

Let us assume that the document ***D*** is as follows:

From an organisational point of view, the structure of the institution is consistent with the principle of hierarchical organisation. Albeit hierarchically structured organisations can be very effective in many cases, it is advisable to consider moving towards a network type organisational model, at the same time keeping consistency.

Step 1. : removal of stopwords,

Step 2. Stemming. E.g., among the remaining words there will be, for example, the words *"consistent"*, *"consistency"*. Users' query can be *"consistent"*, *"consistency",* or other form of this word.

To obtain a common form for user queries and the different word forms in the document, all word forms are/should be transformed to one common form, namely to their lexical root (or stem): "consist".

Porter algorithm (widely used stemming algorithm)

Inverted File Structure

Let $\boldsymbol{E = \{E_1,\ldots,E_j,\ldots,E_m\}}$ denote a set of entities to be searched in a future retrieval system, and let

$$\boldsymbol{D = \{D_1,\ldots,D_j,\ldots,D_m\}}$$

denote the documents corresponding to $\boldsymbol{E}$.

After word identification, stoplisting and stemming, the following set of *terms* is identified:

$$\boldsymbol{T = \{t_1,\ldots,t_i,\ldots,t_n\}.}$$

The set *T* can be used to construct an inverted file structure as follows:

1. Sort the terms $\boldsymbol{t_1,\ldots,t_i,\ldots,t_n}$ alphabetically. For this purpose, some appropriate (fast) sorting algorithm should be used
2. Create an index table *I* in which every row r_i contains exactly one term $\boldsymbol{t_i}$ together with the codes (identifiers) of documents $\boldsymbol{D_j}$ in which that term $\boldsymbol{t_i}$ occurs.

Terms in alphabetical order	Codes of documents in which the term occurs
t_1	$D_{11},\ldots,D_{1k}$
…	
t_i	$D_{i1},\ldots,D_{is}$
…	
t_n	$D_{n1},\ldots,D_{np}$

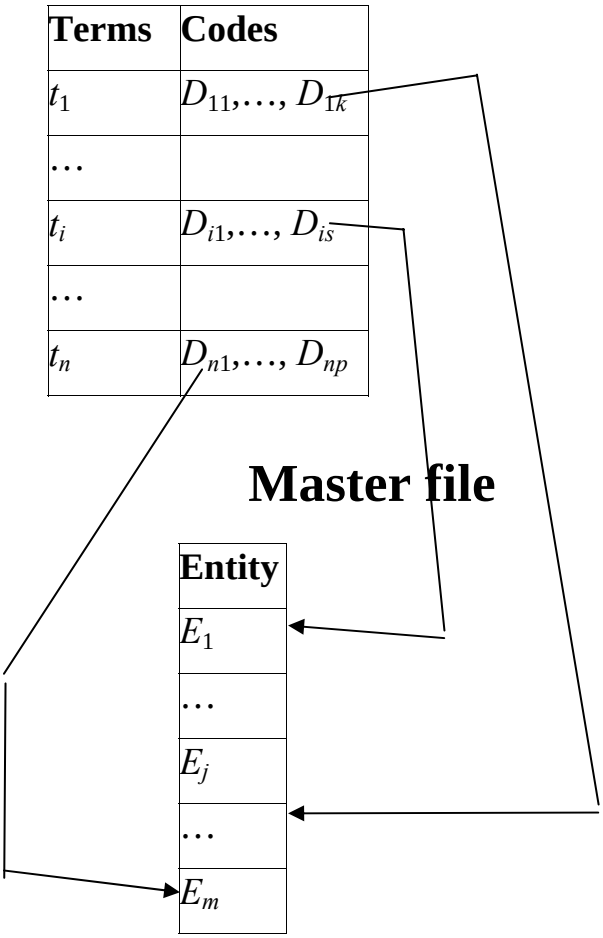
Every document D_j uniquely identifies its corresponding entity E_j

$$\Downarrow$$

a structure *IF* (*Inverted File*) consisting of the index table *I* and of the entities (*master file*) of the set E can be constructed (usually on a disk).

The *Codes* in the index table *I* can also contain the disk addresses of the corresponding entities in the master file.

Index table



The inverted file structure *IF* is used in the following way:

1. Let ***t*** denote a query term. Using appropriate search algorithm, ***t*** is located in the table *I*, i.e., the result of the search is the row:

$$[\boldsymbol{t} \mid \boldsymbol{D_{t1},\ldots,D_{tu}}].$$

2. Using the codes $D_{t1},\ldots,D_{tu}$, the corresponding entities $E_{t1},\ldots,E_{tu}$ can be read from the master file for further processing.

In an inverted file structure, other data can also be stored, such as:

- the number of occurrences of term t_i in document D_j,
- the total number of occurrences of term t_i in all documents,
- etc..

The inverted file structure is a logical structure.

Its physical implementation depends on:

- the properties of the particular computer hardware,
- operating system,
- programming language,
- database management system,
- etc. available.

Term-Document Matrix

Let

$\boldsymbol{E = \{E_1,\ldots,E_j,\ldots,E_m\}}$ denote a set of entities to be searched in a future computerised retrieval system, and

$\boldsymbol{D = \{D_1,\ldots,D_j,\ldots,D_m\}}$ denote the documents corresponding to $\boldsymbol{E}$.

After word identification, stoplisting and stemming the following set of terms has been constructed :

$\boldsymbol{T = \{t_1,\ldots,t_i,\ldots,t_n\}}$.

Construction of term-document matrix *TD*

(*i* = 1,…,*n*, *j* = 1,…,*m*)

1. Establish f_{ij}: the number of times term t_i occurs in document D_j.
2. Construct the *term-document matrix* $TD = (w_{ij})_{n \times m}$, where the entry w_{ij} is referred to as the *weight* of term t_i in the document D_j. The weight is a numerical measure of the extent to which the term reflects the content of the document.

There are several methods to compute the weights.

The mostly used ones are the following:

binary weighting method:

$$w_{ij} = \begin{cases} 1 & if \quad t_i \quad occurs \ in \quad D_j \\ 0 & otherwise \end{cases},$$

frequency weighting method:

$$w_{ij} = f_{ij}.$$

max-tf; max-normalised method:

$$w_{ij} = \frac{f_{ij}}{\max\limits_{1 \le k \le n} f_{kj}} .$$

norm-tf, length-normalized method:

$$w_{ij} = \frac{f_{ij}}{\sqrt{\sum_{k=1}^{n} f_{kj}^2}} .$$

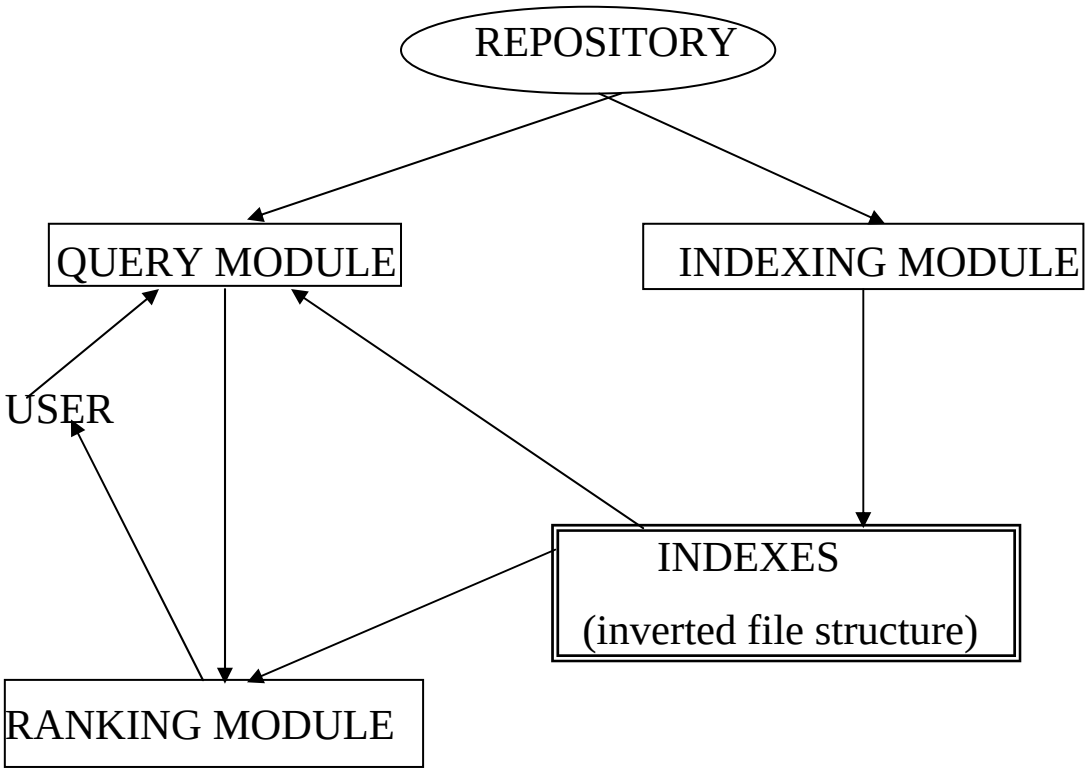
tf-idf, term frequency inverse document frequency method:

$$w_{ij} = f_{ij} \times \left(\log \frac{m}{F_i} \right).$$

norm-tf-idf, length normalized term frequency inverse document frequency method:

$$w_{ij} = \frac{f_{ij} \times \left(\log \frac{m}{F_i} \right)}{\sqrt{\sum_{k=1}^{n} \left(f_{kj} \times \left(\log \frac{m}{F_k} \right) \right)^2}}.$$

General Architecture of a Retrieval System



- **REPOSITORY**.

  The entities (documents) to be searched are stored in a central REPOSITORY (on computer disks). They are collected and entered into the REPOSITORY manually or using specialized computer programs.

- **INDEXING MODULE**.

  Using the documents stored in the REPOSITORY, the INDEXING MODULE creates the INDEXES in the form of inverted file structures. These structures are being used by the QUERY MODULE to find documents that match the user's query.

- **QUERY MODULE**.

  It reads in the user's query. The QUERY MODULE, using INDEXES, finds the documents which match the query (typically, the documents that contain the query terms). It then passes the located documents to the RANKING MODULE.

- **RANKING MODULE**.

  It computes similarity scores (using INDEXES) for the documents located by the QUERY MODULE. Then, the documents are ranked (sorted descendingly) on their similarity score, and are presented to the user in this order (this list is called *hit list*). For the computation of similarity scores, several methods can be used (they are dealt with later on).

Elements of Web Retrieval Technology

World Wide Web

The World Wide Web is a network of electronic documents stored on dedicated computers (*servers*) around the world.

Documents can contain different types of data, such as text, image, or sound. They are stored in units referred to as Web *pages*.

Each page has a unique code, called **URL** (Universal Resource Locator), which identifies its location on a server.

Example,

http://www.dcs.vein.hu/CIR/i2rmeta/i2rmeta.cgi

The number of Web pages is referred to as the *size* of the Web. (More than **12 billion** pages to date).

Major Characteristics of the Web

Most Web documents are in HTML (Hypertext Mark Up Language) format, containing many *tags* (provide important information about the page).

E.g., the tag <b>, which is a bold typeface markup, usually increases the importance of the term it refers to.

Web pages can be less structured (there does not exist a generally recommended or prescribed format)

Also, they are more diverse:

- they can be written in many language, moreover several languages may be used within the same page,
- the grammar of the text in a page may not always be checked very carefully,
- the style used varies to a great extent,
- the length of pages is virtually not limited (if at all, then the limits are posed by, e.g., disk capacity, memory).

Web pages can contain a variety of data types including

- text,
- image,
- sound,
- video,
- executable code.

Many different formats are used, such as

- HTML,
- XML,
- PDF,
- MSWord,
- mp3,
- avi,
- mpeg,
- etc..

While most documents in classical Information Retrieval are considered to be static (e.g., journal papers),

Web pages are dynamic, i.e., they can be

- updated frequently,
- deleted or added,
- dynamically generated.

Web pages can be hyperlinked, which generates a linked network of Web pages. Factors like

- a Universal Resource Locator from a Web page to another page,
- anchor text,
- the underlined, clickable text

can provide additional information about the importance of the target page.

General Architecture of a Web Search Engine

- **CRAWLER MODULE**.
  - In a traditional retrieval system, the documents are stored in a centralised repository, i.e., on computer disks, specifically in a particular institution (university library, computing department in a bank, etc.).
  - As opposed to this, Web pages are stored in a decentralised manner: in computers around the whole world. While this has advantages (e.g., there are no geographic boundaries between documents),

it also means that search engines need to collect documents from around the world.

- This task is being performed by specialised computer programs which together make up the CRAWLER MODULE. They need to run all the time, day and night.
- Virtual robots, named *spiders*, 'walk' on the Web, from page to page, download and send them to the REPOSITORY.

- **REPOSITORY**.

  The Web pages downloaded by spiders are being stored in the REPOSITORY (which physically means computer disks mounted on computers belonging to the company which runs the search engine). Pages are sent from the REPOSITORY to the INDEXING MODULE for further processing. Important or popular pages can be stored for a longer (even a very long) period of time.

- **INDEXING MODULE**.

  The Web pages from the REPOSITORY are being processed by the programs of the INDEXING MODULE (HTML tags are filtered, terms are extracted, etc.). In other words, a compressed representation is obtained for pages by recognising and extracting important information.

- **INDEXES**.

  It is logically organised as an inverted file structure (physically implemented in compressed ways in order to save memory). It is typically divided into several substructures:

  - The ***content structure*** is an inverted structure which stores, for example, terms, anchor text, etc. for pages.
  - The ***link structure*** stores connection information between pages (i.e., which page has a link to which page). The spider may access the link structure to find addresses of uncrawled pages.

- **QUERY MODULE**.

  Step 1. The QUERY MODULE reads in what the user has typed into the query line, analyses and transforms it into an appropriate (for example, numeric code) format.

  Step 2. The QUERY MODULE consults the INDEXES in order to find pages which match the user's query (for example, pages containing the query terms).

  Step 3. It then sends the matching pages to the RANKING MODULE.

- **RANKING MODULE**.

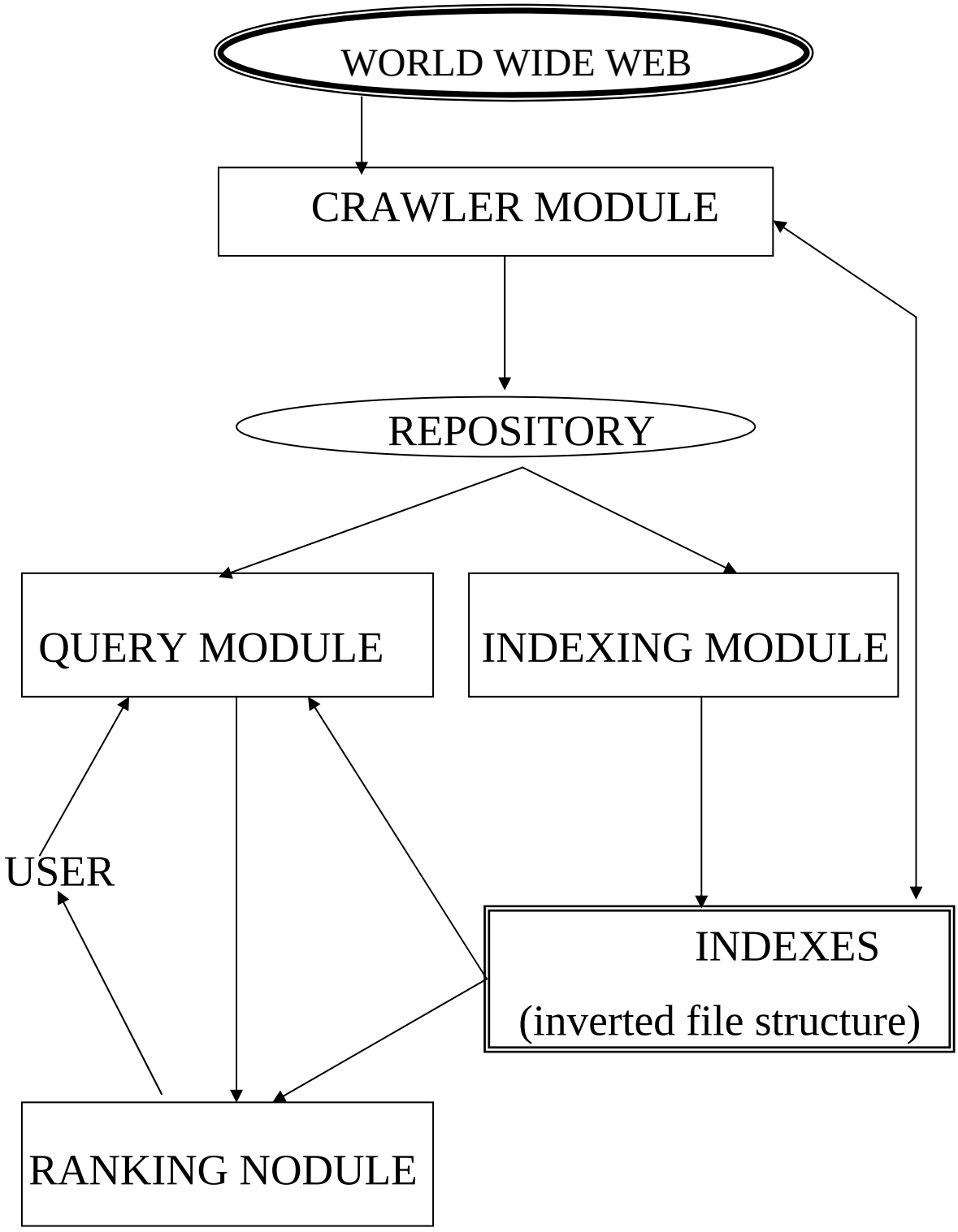
  The pages sent by the QUERY MODULE are ranked (sorted in descending order) according to a similarity score. The list obtained is called *hit list*, and it is presented to the user on the computer screen in the form of a list of URLs.

  The user can access the entire page by clicking on its URL.

The similarity score is computed based on several criteria and using several methods. This calculation is based on a combination of methods from traditional Information Retrieval and Web specific factors.

Typical factors are:

- page content factors (e.g., *tf* in the page),
- on-page factors (e.g., the position of the term in the page, the size of characters of the term),
- link information (which pages link to the page under focus, and which pages it links to),
- etc..


WORLD WIDE WEB
CRAWLER MODULE
REPOSITORY
QUERY MODULE
INDEXING MODULE
USER
INDEXES
(inverted file structure)
RANKING NODULE

General Architecture of

a Web Meta Search Engine

Typically, a meta search engine:

reads in the user’s request,

⇓

sends it to several search engines,

⇓

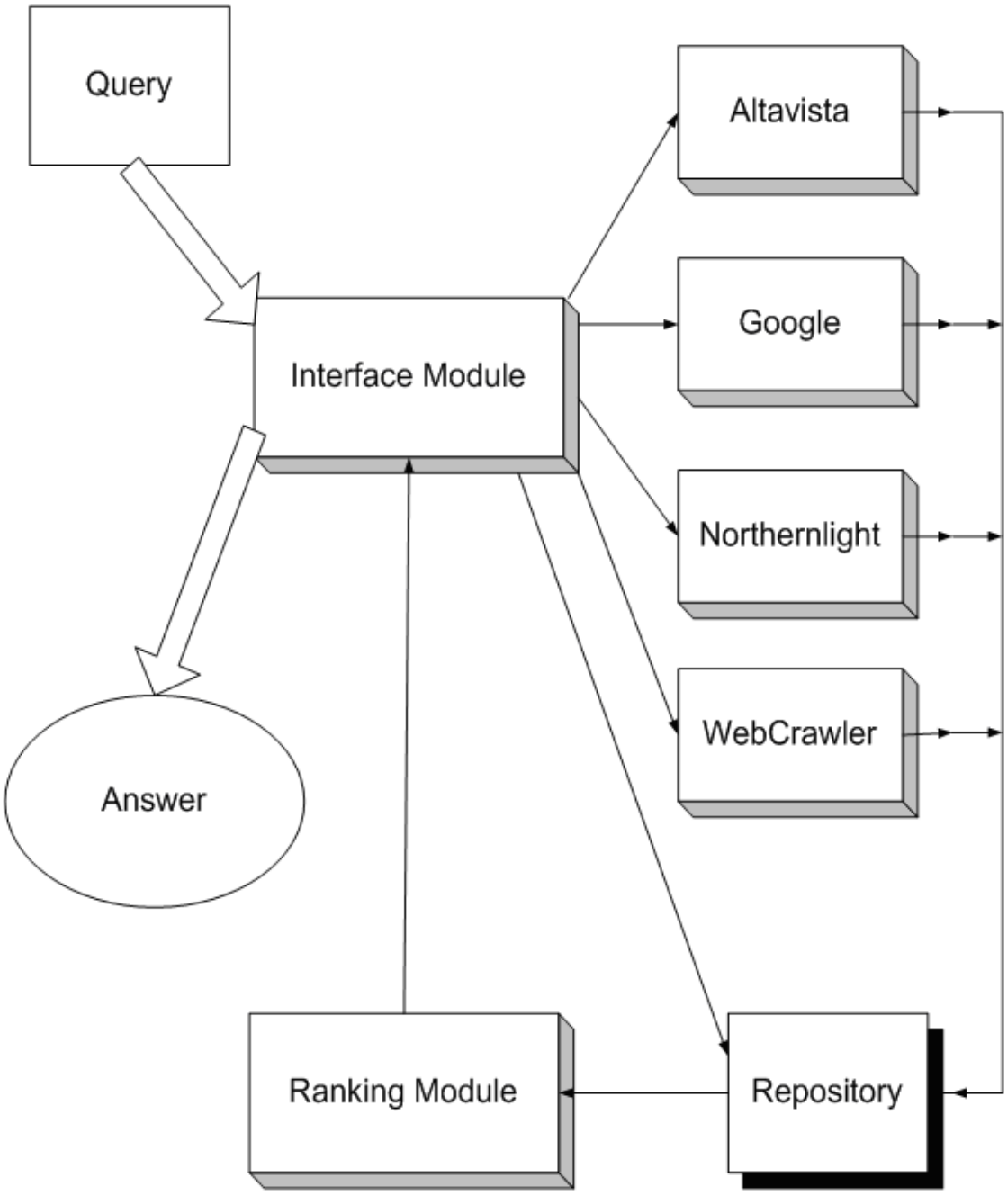
downloads some of the pages

⇓

they return in response to the query

⇓

and then produces its own hit list using those pages.


Query
Altavista
Google
Interface Module
Northernlight
WebCrawler
Answer
Ranking Module
Repository

- **INTERFACE MODULE.**
  - It is written in PERL and works online. The communication with the Web server is performed by CGI.
  - The query is entered as a set of terms (separated by commas), they are Porter-stemmed, and then sent to four commercial spider-based Web search engines as HTTP requests.
  - The first fifty elements from the hit list of each Web search engine are considered, and the corresponding Web pages are downloaded in parallel (Parallel User Agent) for speed.
  - Each Web page undergoes the following processing: tags are removed, terms are identified, stoplisted and Porter-stemmed.
  - The result will be a repository of these pages on the server disk. This repository is processed by the RANKING MODULE.

- **REPOSITORY MODULE.**

  It stores the data sent by the INTERFACE MODULE on the server disk, i.e., the transformed Web pages downloaded by the INTERFACE MODULE. This file is created „on the fly“, during the process of answering the query.

- **RANKING MODULE.**
  - This module is written in C, and works online.
  - Using the query and the Web pages in the repository, it creates a network based on page links as well as terms occurring in both pages and query.
  - The hit list will contain the most important pages, i.e., the pages which are most strongly linked with each other, starting from the query.
  - The hit list is sent to the INTERFACE MODULE which screens it out (answer).