UNIT I

Introduction - History of IR - Components of IR - Issues – Open source Search engine Frameworks - The impact of the web on IR - The role of artificial intelligence (AI) in IR – IR Versus Web Search - Components of a Search engine - Characterizing the Web

INTRODUCTION

Information

- **Cookie Monster's definition:** news or facts about something.

Types of Information

- Text
- XML and structured documents
- Images
- Audio
- Video
- Source Code
- Applications/Web services

Retrieval

- "Fetch something" that's been stored

Information Retieval - Calvin Mooers definition

"Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information." It is the activity of obtaining information resources relevant to an information need from a collection of information resources.

Main objective of IR

- Provide the users with effective access to and interaction with information resources.

Goal of IR

- The goal is to search large document collections to retrieve small subsets relevant to the user's information need.

Purpose/role of an IR system

An information retrieval system is designed to retrieve the documents or information required by the user community. It should make the right information available to the right user. Thus, an information retrieval system aims at collecting and organizing information in one or more subject areas in order to provide it to the user as soon as possible. Thus it serves as a bridge between the world of creators or generators of information and the users of that information.

Application areas within IR

- Cross language retrieval
- Speech/broadcast retrieval
- Text categorization
- Text summarization
- Structured document element retrieval (XML)

Information Retrieval vs Information Extraction

- **Information Retrieval**: Given a set of terms and a set of document terms select only the most relevant document (precision and preferably all the relevant ones (recall).
- **Information Extraction**: Extract from the text what the document means.

Difference between data retrieval and information retrieval

Parameters	**Databases/Data Retrieval**	**Information retrieval**
Example	Data Base Query	WWW Search
What we are retrieving	Structured data	Mostly unstructured
Queries we are posing	Formally defined queries, unambiguous	Expressed in natural language
Matching	Exact	Partial Match, Best Match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic

Kinds of information retrieval systems

Two broad categories of information retrieval system can be identified:

- **In- house**: In- house information retrieval systems are set up by a particular library or information center to serve mainly the users within the organization. One particular type of in-house database is the library catalogue.
- **Online**: Online IR is nothing but retrieving data from web sites, web pages and servers that may include data bases, images, text, tables, and other types.

Features of an information retrieval system

Liston and Schoene suggest that an effective information retrieval system must have provisions for:

- Prompt dissemination of information
- Filtering of information
- The right amount of information at the right time
- Active switching of information
- Receiving information in an economical way
- Browsing
- Getting information in an economical way
- Current literature
- Access to other information systems
- Interpersonal communications, and
- Personalized help.

HISTORY OF IR

- 1960-70's:
  - Initial exploration of text retrieval systems for "small" corpora of scientific abstracts, and law and business documents.
  - Development of the basic Boolean and vector-space models of retrieval.
  - Prof. Salton and his students at Cornell University are the leading researchers in the area.
- 1980's:
  - Large document database systems, many run by companies:
    - Lexis-Nexis
    - Dialog
    - MEDLINE

- 1990's:
  - Searching FTPable documents on the Internet
    - Archie
    - WAIS
  - Searching the World Wide Web
    - Lycos
    - Yahoo
    - Altavista
  - Organized Competitions
    - NIST TREC
  - Recommender Systems
    - Ringo
    - Amazon
    - NetPerceptions
  - Automated Text Categorization & Clustering
- 2000's
  - Link analysis for Web Search
    - Google
  - Automated Information Extraction
    - Whizbang
    - Fetch
    - Burning Glass
  - Question Answering
    - TREC Q/A track
  - Multimedia IR
    - Image
    - Video
    - Audio and music
  - Cross-Language IR
    - DARPA Tides
  - Document Summarization
  - Learning to Rank

IR and Related Areas

1. Database Management
2. Library and Information Science
3. Artificial Intelligence
4. Natural Language Processing
5. Machine Learning

1.Database Management

- Focused on *structured* data stored in relational tables rather than free-form text.
- Focused on efficient processing of well-defined queries in a formal language (SQL).
- Clearer semantics for both data and queries.
- Recent move towards *semi-structured* data (XML) brings it closer to IR.

2.Library and Information Science

- Focused on the human user aspects of information retrieval (human-computer interaction, user interface, visualization).
- Concerned with effective categorization of human knowledge.
- Concerned with citation analysis and *bibliometrics* (structure of information).

- Recent work on *digital libraries* brings it closer to CS & IR.

3.Artificial Intelligence

- Focused on the representation of knowledge, reasoning, and intelligent action.
- Formalisms for representing knowledge and queries:
  - First-order Predicate Logic
  - Bayesian Networks
- Recent work on web ontologies and intelligent information agents brings it closer to IR.

4.Natural Language Processing

- Focused on the syntactic, semantic, and pragmatic analysis of natural language text and discourse.
- Ability to analyze syntax (phrase structure) and semantics could allow retrieval based on *meaning* rather than keywords.

Natural Language Processing: IR Directions

- Methods for determining the sense of an ambiguous word based on context (*word sense disambiguation*).
- Methods for identifying specific pieces of information in a document (*information extraction*).
- Methods for answering specific NL questions from document corpora or structured data like FreeBase or Google's Knowledge Graph.

5.Machine Learning

- Focused on the development of computational systems that improve their performance with experience.
- Automated classification of examples based on learning concepts from labeled training examples (*supervised learning*).
- Automated methods for clustering unlabeled examples into meaningful groups (*unsupervised learning*).

Machine Learning: IR Directions

- Text Categorization
  - Automatic hierarchical classification (Yahoo).
  - Adaptive filtering/routing/recommending.
  - Automated spam filtering.
- Text Clustering
  - Clustering of IR query results.
  - Automatic formation of hierarchies (Yahoo).
- Learning for Information Extraction
- Text Mining
- Learning to Rank

COMPONENTS OF INFORMATION RETRIEVAL

Information retrieval is concerned with representing, searching, and manipulating large collections of electronic text and other human-language data.

Basic IR System Architecture (Stefan Buettcher)

Figure illustrates the major components in an IR system. Before conducting a search, a user has an **information need**, which underlies and drives the search process. This information need is sometimes referred as a **topic**, particularly when it is presented in written form as part of a text collection for IR evaluation. As a result of the information need, the **user** constructs and issues a **query** to the IR system. This query consists of a smaller number of terms, with two or three terms being typical for a Web search. Depending on the information need, a query term may be a date, a number, a musical note, or a phrase. Wildcard operators and other partial-match operators may also be permitted in query terms. For example, the term "inform*" might match any word starting with that prefix ("inform", "informs", "informal", "informant", informative", etc.).

Although users typically issue simple keyword queries, IR systems often support a richer query syntax, frequently with complex Boolean and pattern matching operators. These facilities may be used to limit a search

to a particular Web site, to specify constraints on fields such as author and title, or to apply other filters, restricting the search to a subset of the collection. A user interface mediates between the user and the IR system, simplifying the query creation process when these richer query facilities are required.

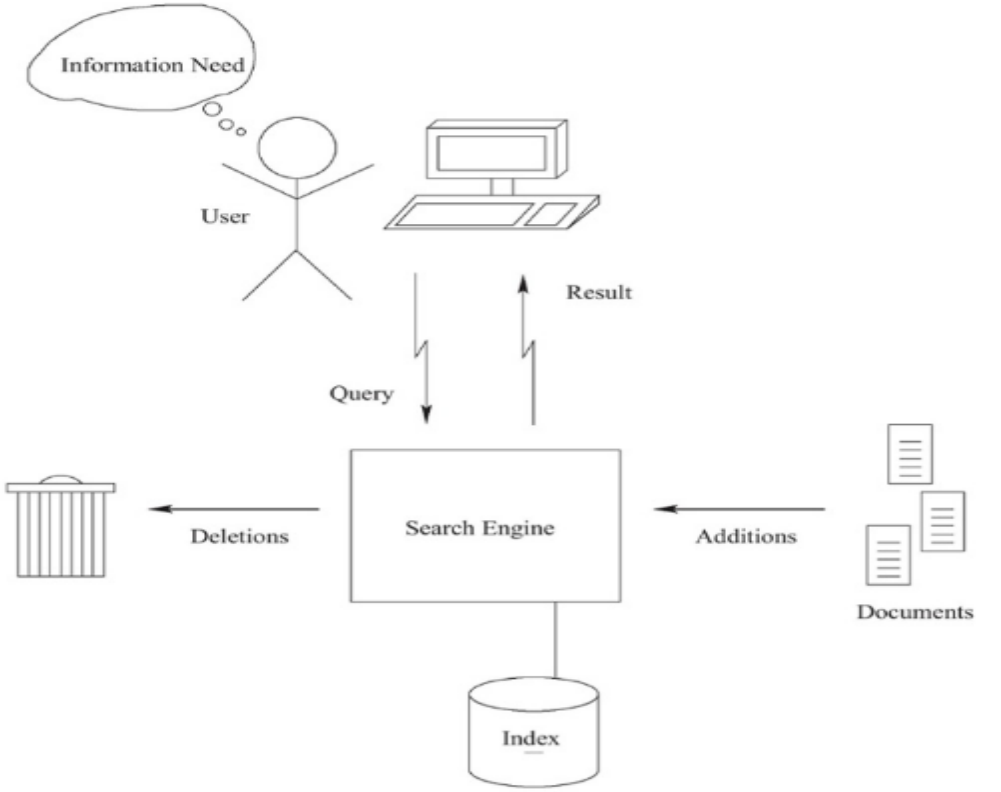


Figure 1.1 Components of an IR system.

The user's query is processed by a **search engine**, which may be running on the user's local machine, on a large cluster of machines in a remote geographic location, or anywhere in between. A major task of a search engine is to maintain and manipulate an **inverted index** for a **document collection**. This index forms the principal data structure used by the engine for searching and relevance ranking. As its basic function, an inverted index provides a mapping between terms and the locations in the collection in which they occur.

To support relevance ranking algorithms, the search engine maintains collection statistics associated with the index, such as the number of documents containing each term and the length of each document. In addition the search engine usually has access to the original content of the documents in order to report meaningful results back to the user.

Using the inverted index, collection statistics, and other data, the search engine accepts queries from its users, processes these queries, and returns ranked lists of results. To perform relevance ranking, the search engine computes a score, sometimes called a retrieval status value (RSV), for each document. After sorting documents according to their scores, the result list must be subjected to further processing, such as the removal of duplicate or redundant results. For example, a Web search engine might report only one or results from a single host or domain, eliminating the others in favor of pages from different sources.

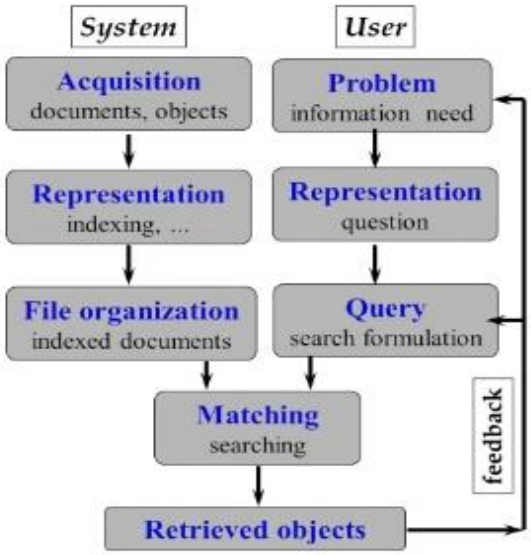
Traditional IRS

Three major components of Traditional IRS

1. Document subsystem
    a) Acquisition
    b) Representation
    c) File organization
2. User sub system
    a) Problem

b) Representation
c) Query

3. Searching /Retrieval subsystem
   a) Matching
   b) Retrieved objects



An information retrieval system thus has three major components- the document subsystem, the users subsystem, and the searching/retrieval subsystem. These divisions are quite broad and each one is designed to serve one or more functions, such as:

- Analysis of documents and organization of information (creation of a document database)
- Analysis of user's queries, preparation of a strategy to search the database
- Actual searching or matching of users queries with the database, and finally
- Retrieval of items that fully or partially match the search statement.

Acquisition (Document subsystem)

- Selection of documents & other objects from various web resources.
- Mostly text based documents
  - full texts, titles, abstracts...
  - but also other objects:
    - data, statistics, images, maps, trade marks, sounds ...
- The data are collected by web crawler and stored in data base.

Representation of documents, objects(document subsystem)

- Indexing – many ways :
  - free text terms (even in full texts)
  - controlled vocabulary - thesaurus
  - manual& automatic techniques.
- Abstracting; summarizing
- Bibliographic description:
  - author, title, sources, date…
  - metadata
- Classifying, clustering
- Organizing in fields & limits
  - Basic Index, Additional Index. Limits

File organization (Document subsystem)

- Sequential

    - record (document) by record
- Inverted
    - term by term; list of records under each term
- Combination
- indexes inverted, documents sequential
- When citation retrieved only, need for document files
- Large file approaches
- for efficient retrieval by computers

Problem (user subsystem)

- Related to users' task, situation
    - vary in specificity, clarity
- Produces information need
    - ultimate criterion for effectiveness of retrieval
        - how well was the need met?
- Information need for the same problem may change, evolve, shift during the IR process adjustment in searching
    - often more than one search for same problem over time

Representation (user subsystem)

- Converting a concept to query.
- What we search for.
- These are stemmed and corrected using dictionary.
- Focus toward a good result
- Subject to feedback changes

Query - search statement (user & system)

- Translation into systems requirements & limits
    - start of human-computer interaction
        - query is the thing that goes into the computer
- Selection of files, resources
- Search strategy - selection of:
    - search terms & logic
    - possible fields, delimiters
    - controlled & uncontrolled vocabulary
    - variations in effectiveness tactics
- Reiterations from feedback
    - several feedback types: relevance feedback, magnitude feedback..
    - query expansion & modification

Matching - searching (Searching subsystem)

- Process of matching, comparing
    - search: what documents in the file match the query as stated?
- Various search algorithms:
    - exact match - Boolean
        - still available in most, if not all systems
    - best match - ranking by relevance
        - increasingly used e.g. on the web
    - hybrids incorporating both
        - e.g. Target, Rank in DIALOG
- Each has strengths, weaknesses
    - No 'perfect' method exists and probably never will

Retrieved documents -from system to user (IR Subsystem)

- Various order of output:
  - Last In First Out (LIFO); sorted
  - ranked by relevance
  - ranked by other characteristics
- Various forms of output
- When citations only: possible links to document delivery
- Base for relevance, utility evaluation by users
- Relevance feedback

ISSUES IN IR (Bruce Croft)

Information retrieval is concerned with representing, searching, and manipulating large collections of electronic text and other human-language data.

Three Big Issues in IR

1.Relevance

- It is the fundamental concept in IR.
- A relevant document contains the information that a person was looking for when she submitted a query to the search engine.
- There are many factors that go into a person's decision as to whether a document is relevant.
- These factors must be taken into account when designing algorithms for comparing text and ranking documents.
- Simply comparing the text of a query with the text of a document and looking for an exact match, as might be done in a database system produces very poor results in terms of relevance.

To address the issue of relevance, **retrieval models** are used.

- A retrieval model is a formal representation of the process of matching a query and a document. It is the basis of the ranking algorithm that is used in a search engine to produce the ranked list of documents.
- A good retrieval model will find documents that are likely to be considered relevant by the person who submitted the query.
- The retrieval models used in IR typically model the statistical properties of text rather than the linguistic structure. For example, the ranking algorithms are concerned with the counts of word occurrences than whether the word is a noun or an adjective.

2.Evaluation

- Two of the evaluation measures are precision and recall.
  Precision is the proportion of retrieved documents that are relevant.
  Recall is the proportion of relevant documents that are retrieved.

$$\text{Precision} = \frac{\text{Relevant documents} \cap \text{Retrieved documents}}{\text{Retrieved documents}}$$

$$\text{Recall} = \frac{\text{Relevant documents} \cap \text{Retrieved documents}}{\text{Relevant documents}}$$

- When the recall measure is used, there is an assumption that all the relevant documents for a given query are known. Such an assumption is clearly problematic in a web search environment, but with smaller test collection of documents, this measure can be useful. It is not suitable for large volumes of log data.

3.Emphasis on users and their information needs

- The users of a search engine are the ultimate judges of quality. This has led to numerous studies on how people interact with search engines and in particular, to the development of techniques to help people express their information needs.
- Text queries are often poor descriptions of what the user actually wants compared to the request to a database system, such as for the balance of a bank account.

- Despite their lack of specificity, one-word queries are very common in web search. A one-word query such as "cats" could be a request for information on where to buy cats or for a description of the Cats (musical).
- Techniques such as query suggestion, query expansion and relevance feedback use interaction and context to refine the initial query in order to produce better ranked results.

Main problems

- Document and query indexing
  - How to represent their contents?
  - Query evaluation
- To what extent does a document correspond to a query?
  - System evaluation
  - How good is a system?
  - Are the retrieved documents relevant? (precision)
  - Are all the relevant documents retrieved? (recall)

Why is IR difficult?

- Vocabularies mismatching
  - The language can be used to express the same concepts in many different ways, with different words. This is referred to as the vocabulary mismatch problem in information retrieval.
  - E.g. Synonymy: car vs automobile
- Queries are ambiguous
- Content representation may be inadequate and incomplete
- The user is the ultimate judge, but we don't know how the judge judges.

Challenges in IR

- Scale, distribution of documents
- Controversy over the unit of indexing
- High heterogeneity
- Retrieval strategies

OPEN SOURCE SEARCH ENGINE FRAMEWORKS

Open source

Open source software is software whose source code is available for modification or enhancement by anyone. "Source code" is the part of software that most computer users don't ever see; it's the code computer programmers can manipulate to change how a piece of software—a "program" or "application"—works. Programmers who have access to a computer program's source code can improve that program by adding features to it or fixing parts that don't always work correctly.

Advantage of open source

- The right to use the software in any way.
- There is usually no license cost and free of cost.
- The source code is open and can be modified freely.
- Open standards.
- It provides higher flexibility.

Disadvantage of open source

- There is no guarantee that development will happen.
- It is sometimes difficult to know that a project exist, and its current status.
- No secured follow-up development strategy.

Closed software

Closed software is a term for software whose license does not allow for the release or distribution of the software's source code. Generally it means only the binaries of a computer program are distributed and the

license provides no access to the programs source code. The source code of such programs is usually regarded as a trade secret of the company. Access to source code by third parties commonly requires the party to sign a non-disclosure agreement.

Search Engine

A search engine is a document retrieval system design to help find information stored in a computer system, such as on the WWW. The search engine allows one to ask for content meeting specific criteria and retrieves a list of items that match those criteria.

Lists of open source search engines

1. Apache Lucene
2. Sphinx
3. Whoosh
4. Carrot^2

1.Apache Lucene Core

Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.

Powerful features through a simple API:

- Scalable, High-Performance Indexing
- Over 150GB/hour on modem hardware
- small RAM requirements -- only 1MB heap
- incremental indexing as fast as batch indexing
- index size roughly 20-30% the size of text indexed
- Powerful, Accurate and Efficient Search Algorithms
- ranked searching -- best results returned first
- many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more
- fielded searching (e.g. title, author, contents)
- sorting by any field
- multiple-index searching with merged results
- allows simultaneous update and searching
- flexible faceting, highlighting, joins and result grouping
- fast, memory-efficient and typo-tolerant suggesters
- pluggable ranking models, including the Vector Space Model and Okapi BM25
- configurable storage engine (codecs)

Highlights of this Lucene release include:

- Numerous improvements to LatLonPoint, for indexing a latitude/longitude point and searching by polygon, distance or box, or finding nearest neighbors
- Geo3D now has simple APIs for creating common shape queries, matching LatLonPoint
- Faster indexing and searching of points.
- Faster geo-spatial indexing and searching for LatLonPoint, Geo3D and GeoPoint
- Hardlink Copy Directory Wrapper optimizes file copies using hard links
- In case of contention, the query cache now prefers returning an uncached Scorer rather than waiting on a lock.

Highlights of this Solar release include:

- Added graph traversal support, and new "sort" and "random" streaming expressions. It's also now possible to create streaming expressions with the Solr Admin UI.
- Fixed the ENUM faceting method to not be unnecessarily rewritten to FCS, which was causing slowdowns.
- Reduced garbage creation when creating cache entries.

- New [subquery] document transformer to obtain related documents per result doc.
- EmbeddedSolrServer allocates heap much wisely even with plain document list without callbacks.
- New GeoJSON response writer for encoding geographic data in query responses.

2.Sphinx

Sphinx is a full-text search engine, publicly distributed under GPL version 2. Technically, Sphinx is a standalone software package provides fast and relevant full-text search functionality to client applications. It was specially designed to integrate well with SQL databases storing the data, and to be easily accessed by scripting languages. However, Sphinx does not depend on nor require any specific database to function.

Applications can access Sphinx search daemon (searchd) using any of the **three different access methods**:

a) via Sphinx own implementation of MySQL network protocol (using a small SQL subset called SphinxQL, this is recommended way)

b) via native search API (SphinxAPI) or

c) via MySQL server with a pluggable storage engine (SphinxSE).

Starting from version 1.10-beta, Sphinx supports **two different indexing backends**:

a) **"Disk" index backend** - Disk indexes support online full-text index rebuilds, but online updates can only be done on non-text (attribute) data.

b) **"Realtime" (RT) index backend** - RT indexes additionally allow for online full-text index updates. Previous versions only supported disk indexes.

Data can be loaded into disk indexes using a so-called data source. Built-in sources can fetch data directly from MySQL, PostgreSQL, MSSQL, ODBC compliant database (Oracle, etc) or a pipe in TSV or a custom XML format. Adding new data sources drivers (eg. to natively support other DBMSes) is designed to be as easy as possible. RT indexes, as of 1.10-beta, can only be populated using SphinxQL. As for the name, Sphinx is an acronym which is officially decoded as SQL Phrase Index.

Sphinx features

Key Sphinx features are:

- high indexing and searching performance;
- advanced indexing and querying tools;
- advanced result set post-processing (SELECT with expressions, WHERE, ORDER BY, GROUP BY, HAVING etc over text search results);
- proven scalability up to billions of documents, terabytes of data, and thousands of queries per second;
- easy integration with SQL and XML data sources, and SphinxQL, SphinxAPI, or SphinxSE search interfaces;
- easy scaling with distributed searches.

To expand a bit, Sphinx:

- has high indexing speed (upto 10-15 MB/sec per core on an internal benchmark);
- has high search speed (upto 150-250 queries/sec per core against 1,000,000 documents, 1.2 GB of data on an internal benchmark);
- has high scalability (biggest known cluster indexes over 3,000,000,000 documents, and busiest one peaks over 50,000,000 queries/day);
- provides good relevance ranking through combination of phrase proximity ranking and statistical (BM25) ranking;
- provides distributed searching capabilities;
- provides document excerpts (snippets) generation;
- provides searching from within application with SphinxQL or SphinxAPI interfaces, and from within MySQL with pluggable SphinxSE storage engine;
- supports boolean, phrase, word proximity and other types of queries;
- supports multiple full-text fields per document (upto 32 by default);

- supports multiple additional attributes per document (ie. groups, timestamps, etc);
- supports stopwords;
- supports morphological word forms dictionaries;
- supports tokenizing exceptions;
- supports UTF-8 encoding;
- supports stemming (stemmers for English, Russian, Czech and Arabic are built-in; and stemmers for French, Spanish, Portuguese, Italian, Romanian, German, Dutch, Swedish, Norwegian, Danish, Finnish, Hungarian, are available by building third party libstemmer library);
- supports MySQL natively (all types of tables, including MyISAM, InnoDB, NDB, Archive, etc are supported);
- supports PostgreSQL natively;
- supports ODBC compliant databases (MS SQL, Oracle, etc) natively;

Performance and scalability

- Indexing speed of up to 10-15 MB/sec per core and HDD.
- Searching speed of over 500 queries/sec against 1,000,000 document/1.2 GB collection using a 2-core desktop system with 2 GB of RAM
- The biggest known installation using Sphinx, Boardreader.com, indexes 16 billion documents
- The busiest known installation, Craigslist, serves over 300,000,000 queries/dayand more than 50 billion page views/month

3.Whoosh

Whoosh was created by Matt Chaput. It started as a quick and dirty search server for the online documentation of the Houdini 3D animation software package. Whoosh is a fast, featureful full-text indexing and searching library implemented in pure Python. Programmers can use it to easily add search functionality to their applications and websites. Every part of how Whoosh works can be extended or replaced to meet your needs exactly.

Whoosh's features include:

- Pythonic API.
- Pure-Python. No compilation or binary packages needed, no mysterious crashes.
- Fielded indexing and search.
- Fast indexing and retrieval – faster than any other pure-Python, scoring, full-text search solution I know of.
- Pluggable scoring algorithm (including BM25F), text analysis, storage, posting format, etc.
- Powerful query language.
- Pure Python spell-checker (as far as I know, the only one).

Whoosh might be **useful** in the following circumstances:

- Anywhere a pure-Python solution is desirable to avoid having to build/compile native libraries.
- As a research platform
- When an easy-to-use Pythonic interface is more important to you than raw speed.

4.Carrot^2 Search Engine

Carrot^2 is an Open Source Search Results Clustering Engine. It can automatically organize small collections of documents (search results but not only) into thematic categories.

The architecture of Carrot² is based on *processing components* arranged into pipelines. Two major groups or processing components in Carrot² are:

a)Document sources

b)Clustering algorithms

a)Document sources

Document sources provide data for further processing. Typically, they would e.g. fetch search results from an external search engine, Lucene / Solr index or load text files from a local disk.

Currently, Carrot² has built-in support for the following document sources:

- Bing Search API
- Lucene index
- OpenSearch
- PubMed
- Solr server
- eTools metasearch engine
- Generic XML files

Other document sources can be integrated based on the code examples provided with Carrot² distribution.

b)Clustering algorithms

Carrot offers two specialized document clustering algorithms that place emphasis on the quality of cluster labels:

- Lingo a clustering algorithm based on the Singular value decomposition
- STC Suffix Tree Clustering

Other algorithms can be easily added to Carrot.

Tools

Carrot offers a number of supporting tools that can be used to quickly set up clustering on custom data, further tuning of clustering results and exposing Carrot² clustering as a remote service:

1. **Carrot² Document Clustering Workbench**: a standalone GUI application for experimenting with Carrot clustering on data from common search engines or custom data
2. **Carrot² Document Clustering Server**: exposes Carrot² clustering as a REST service
3. **Carrot² Command Line Interface**: applications that allow invoking Carrot² clustering from command line
4. **Carrot² Web Application**: exposes Carrot² clustering as a web application for end users.

THE IMPACT OF WEB ON IR (Ricardo Baeza)

Tim Berners-Lee conceived the conceptual Web in 1989, tested it successfully in December of 1990, and released the first Web server early in 1991. It was called World Wide Web, and is referred as Web. At that time, no one could have imagined the impact that the Web would have. The Web boom, characterized by exponential growth on the volume of data and information, imply that various daily tasks such as e-commerce, banking, research, entertainment, and personal communication can no longer be done outside the Web if convenience and low cost are to be granted.

The amount of textual data available on the Web is estimated in the order of petabytes. In addition, other media, such as images, audio, and video, are also available in even greater volumes. Thus, the Web can be seen as a very large, public and unstructured but ubiquitous data repository, which triggers the need for efficient tools to manage, retrieve, and filter information from the Web. As a result, Web search engines have become one of the most used tools in the Internet. Additionally, information finding is also becoming more important in large Intranets, in which one might need to extract or infer new information to support a decision process, a task called data mining (or Web mining for the particular case of the Web).

The very large volume of data available, combined with the fast pace of change, make the retrieval of relevant information from the Web a really hard task. To cope with the fast pace of change, efficient crawling of the Web has become essential. In spite of recent progress in image and non-textual data search in general, the existing techniques do not scale up well on the Web. Thus, search on text continues to be the most popular and most studied alternative. While most search engines are based in the United States and focus on documents in English, there are important non-US search engines that have been designed for specific languages and can

handle various scripts and alphabets, such as Kanji or Cyrillic. Examples include Baidu in China, Yandex in Russia, and Naver in South Korea.

The search continues to be dominated by a "syntactic" paradigm in which documents that contain user specified words or patterns are retrieved in response to a query. An alternative approach to syntactic search is to do a natural language analysis of the text. Techniques to preprocess natural language and extract the text semantics have been around for a while, yet they are not very effective. In fact, except for some fast entity extraction tools that have been devised recently, these techniques are too costly to be used with large amounts of data. In addition, in most cases they are only effective with well written and structured text, in combination with a thesaurus, or other contextual information such as a specific knowledge domain.

There are basically **two main forms of exploring the Web**.

- Issue a word-based query to a search engine that indexes a portion of the Web documents.
- Browse the Web, which can be seen as a sequential search process of following hyperlinks, as embodied for example, in Web directories that classify selected Web documents by subject.

Additional methods exist, such as taking advantage of the hyperlink structure of the Web, yet they are not fully available, and likely less well known and also much more complex.

A Challenging Problem

Let us now consider the main challenges posed by the Web with respect to search. We can divide them in two classes: those that relate to the data itself, which we refer to as data-centric, and those that relate to the users and their interaction with the data, which we refer as interaction-centric.

Data-centric challenges

- **Distributed data**. Due to the intrinsic nature of the Web, data spans over a large number of computers and platforms. These computers are interconnected with no predefined topology and the available bandwidth and reliability on the network interconnections vary widely.
- **High percentage of volatile data**. Due to Internet dynamics, new computers and data can be added or removed easily. To illustrate, early estimations showed that 50% of the Web changes in a few months. Search engines are also confronted with dangling (or broken) links and relocation problems when domain or file names change or disappear.
- **Large volume of data**. The fast growth of the Web poses scaling issues that are difficult to cope with, as well as dynamic Web pages, which are in practice unbounded.
- **Unstructured and redundant data**. The Web is not a huge distributed hypertext system, as some might think, because it does not follow a strict underlying conceptual model that would guarantee consistency. Indeed, the Web is not well structured either at the global or at the individual HTML page level. HTML pages are considered by some as semi-structured data in the best case. Moreover, a great deal of Web data are duplicated either loosely (such as the case of news originating from the same news wire) or strictly, via mirrors or copies. Approximately 30% of Web pages are (near) duplicates. Semantic redundancy is probably much larger.
- **Quality of data**. The Web can be considered as a new publishing media. However, there is, in most cases, no editorial process. So, data can be inaccurate, plain wrong, obsolete, invalid, poorly written or, as if often the case, full of errors, either innocent (typos, grammatical mistakes, OCR errors, etc.) or malicious. Typos and mistakes, specially in foreign names are pretty common.
- **Heterogeneous data**. Data not only originates from various media types, each coming in different formats, but it is also expressed in a variety of languages, with various alphabets and scripts (e.g. India), which can be pretty large (e.g. Chinese or Japanese Kanji).

Many of these challenges, such as the variety of data types and poor data quality, cannot be solved by devising better algorithms and software, and will remain a reality simply because they are problems and issues (consider, for instance, language diversity) that are intrinsic to human nature.

Interaction-centric challenges

- **Expressing a query**. Human beings have needs or tasks to accomplish, which are frequently not easy to express as "queries". Queries, even when expressed in a more natural manner, are just a reflection of

information needs and are thus, by definition, imperfect. This phenomenon could be compared to Plato's cave metaphor, where shadows are mistaken for reality.

- **Interpreting results**. Even if the user is able to perfectly express a query, the answer might be split over thousands or millions of Web pages or not exist at all. In this context, numerous questions need to be addressed. Examples include: How do we handle a large answer? How do we rank results? How do we select the documents that really are of interest to the user? Even in the case of a single document candidate, the document itself could be large. How do we browse such documents efficiently?

In the current state of the Web, search engines need to deal with plain HTML and text, as well as with other data types, such as multimedia objects, XML data and associated semantic information, which can be dynamically generated and are inherently more complex.

If the Semantic Web becomes a reality and overcomes all its inherent social issues, an XML-based Web, with standard semantic metadata and schema, might become available. In this hypothetical world, IR would become easier, and even multimedia search would be simplified. Spam would be much easier to avoid as well, as it would be easier to recognize good content. On the other hand, new retrieval problems would appear, such as XML processing and retrieval, and Web mining on structured data, both at a very large scale.

THE ROLE OF ARTIFICIAL INTELLIGENCE (AI) IN IR

Information Retrieval

- The amount of available information is growing at an incredible rate, for example the Internet and World Wide Web.
- Information are stored in many forms e.g. images, text, video, and audio.
- Information Retrieval is a way to separate relevant data from irrelevant.
- IR field has developed successful methods to deal effectively with huge amounts of information.
    - Common methods include the Boolean, Vector Space and Probabilistic models.

Artificial Intelligence

- Study of how to construct intelligent machines and systems that can simulate or extend the development of human intelligence.

Integration of Artificial Intelligence and Information Retrieval

- Both IR and AI fields are developed in parallel during the early days of computers.
- The fields of artificial intelligence and information retrieval share a common interest in developing more capable computer systems.
- The integration of Artificial Intelligence and Information Retrieval has led to the following development:
    - Development of methods to learn user's information needs.
    - Extract information based on what has been learned.
    - Represent the semantics of information.

What is Intelligence?

According to Cook et.al.

- Acquisition: the ability to acquire new knowledge.
- Automatization: the ability to refine procedures for dealing with a novel situation into an efficient functional form.
- Comprehension: the ability to know, understand, and deal with novel problems.
- Memory management: the ability to represent knowledge in memory, to map knowledge on to that memory representation, and to access the knowledge in memory.
- Metacontrol: the ability to control various processes in intelligent behavior.
- Numeric ability: the ability to perform arithmetic operations.
- Reasoning: the ability to use problem-solving knowledge.
- Social competence: the ability to interact with and understand other people, machines or programs.

- Verbal perception: the ability to recognize natural language.
- Visual perception: the ability to recognize visual images.

What are Intelligent IR Systems?

- The concept of 'intelligent' information retrieval was first suggested in the late 1970s.
- Not pursued by IR Community until early 1990s.

Definitions

- An intelligent IR system can simulate the human thinking process on information processing and intelligence activities to achieve information and knowledge storage, retrieval and reasoning, and to provide intelligence support.
- In an Intelligent IR system, the functions of the human intermediary are performed by a program, interacting with the human user.
- Intelligent IR is performed by a computer program (intelligent agent), which acts on (minimal or no explicit) instructions from a human user, retrieves and presents information to the user without any other interaction.

How to introduce AI into IR systems?

- Can't take the "Human Factor" completely out of the equation.
    - A program which takes a query as input, and returns documents as output, without affording the opportunity for judgment, modification and especially interaction with text, or with the program, is one which would not qualify as an IR system at all.
    - Ignores user interaction and relevance feedback.
- Some processes which can't be performed by any other component than the user.
    - Judgment" is a process which can only be performed by the user.
- The question is, "where" should AI be introduced into the IR system?
- Levels of user and system involvement, according to Bates '90:
    - Level 0 – No system involvement (User comes up with a tactic, formulating a query, coming up with a strategy and thinking about the outcome)
    - Level 1 – User can ask for information about searching (System suggests tactics that can be used to formulate queries e.g. help)
    - Level 2 – User simply enters a query, suggests what needs to be done, and the system executes the query to return results.
    - Level 3 – First signs of AI. System actually starts suggesting improvements to user.
    - Level 4 – Full Automation. User queries are entered and the rest is done by the system.

Barriers to Intelligent IR Systems

- Common Sense Reasoning.
- Natural Language Processing.
- Knowledge Acquisition, Representation, and Maintenance.
- Difficulty in Scaling Up Prototypes to Operational Systems.
- Level of Effort, Technical Expertise, and Expense.

Some AI methods currently used in Intelligent IR Systems

- Web Crawlers (for information extraction)
- Mediator Techniques (for information integration)
- Ontologies (for intelligent information access by making semantics of information explicit and machine readable)
- Neural Networks (for document clustering & preprocessing)
    - Kohonen Neural Networks - Self Organizing maps
    - Hopefield Networks
    - Semantic Networks

Neural Networks in IR

- Based on Neural Networks

- Document clustering can be viewed as classification in document*document space
- Thesaurus construction can be viewed as laying out a coordinate system in the index*index space
- Indexing itself can be viewed as mappings in the document*index space
- Searching can be conceptualized as connections and activations in the index*document space

- Applying Neural Networks to Information Retrieval will likely produce information systems that will be able to:
  - recall memories despite failed individual memory units
  - modify stored information in response to new inputs from the user
  - retrieve "nearest neighbor" data when no exact data match exists
  - associatively recall information despite noise or missing pieces in the input
  - categorize information by their associative patterns

Conclusion

- AI offers us a powerful set of tools, especially when they are combined with conventional and other innovative computing tools. However, it is not an easy task to master those tools and employ them skillfully to build truly significant intelligent systems.
- By recognizing the limitations of modern artificial intelligence techniques, we can establish realistic goals for intelligent information retrieval systems and devise appropriate system development strategies.
- AI models like the neural network will probably not replace traditional IR approaches anytime soon. However, the application of neural network models can make an IR system more powerful.

COMPARING WEB SEARCH TO TRADITIONAL INFORMATION RETRIEVAL

IR Versus Web Search (Mark Levene)

Traditional IR systems normally index a closed collection of documents, which are mainly text-based and usually offer little linkage between documents. Traditional IR systems are often referred to as *full-text retrieval systems*. Libraries were among the first to adopt IR to index their catalogs and later, to search through information which was typically imprinted onto CD-ROMs. The main aim of traditional IR was to return relevant documents that satisfy the user's information need. Although the main goal of satisfying the user's need is still the central issue in web IR (or web search), there are some very specific challenges that web search poses that have required new and innovative solutions.

- The first important difference is the scale of web search, as we have seen that the current size of the web is approximately 600 billion pages. This is well beyond the size of traditional document collections.
- The Web is dynamic in a way that was unimaginable to traditional IR in terms of its rate of change and the different types of web pages ranging from static types (HTML, portable document format (PDF), DOC, Postscript, XLS) to a growing number dynamic pages written in scripting languages such a JSP, PHP or Flash. We also mention that a large number of images, videos, and a growing number of programs are delivered through the Web to our browsers.
- The Web also contains an enormous amount of duplication, estimated at about 30%. Such redundancy is not present in traditional corpora and makes the search engine's task even more difficult.
- The quality of web pages vary dramatically; for example, some web sites create web pages with the sole intention of manipulating the search engine's ranking, documents may contain misleading information, the information on some pages is just out of date, and the overall quality of a web page may be poor in terms of its use of language and the amount of useful information it contains. The issue of quality is of prime importance to web search engines as they would very quickly lose their audience if, in the top-ranked positions, they presented to users poor quality pages.
- The range of topics covered on the Web is completely open, as opposed to the closed collections indexed by traditional IR systems, where the topics such as in library catalogues, are much better defined and constrained.
- Another aspect of the Web is that it is globally distributed. This poses serious logistic problems to search engines in building their indexes, and moreover, in delivering a service that is being used from all over the globe. The sheer size of the problem is daunting, considering that users will not tolerate anything but

an immediate response to their query. Users also vary in their level of expertise, interests, information-seeking tasks, the language(s) they understand, and in many other ways.

- Users also tend to submit short queries (between two to three keywords), avoid the use of anything but the basic search engine syntax, and when the results list is returned, most users do not look at more than the top 10 results, and are unlikely to modify their query. This is all contrary to typical usage of traditional IR.
- The hypertextual nature of the Web is also different from traditional document collections, in giving users the ability to surf by following links.
- On the positive side (for the Web), there are many roads (or paths of links) that "lead to Rome" and you need only find one of them, but often, users lose their way in the myriad of choices they have to make.
- Another positive aspect of the Web is that it has provided and is providing impetus for the development of many new tools, whose aim is to improve the user's experience.

	Classical IR	**Web IR**
Volume	Large	Huge
Data quality	Clean, No duplicates	Noisy, Duplicates available
Data change rate	Infrequent	In flux
Data Accessibility	Accessible	Partially accessible
Format diversity	Homogeneous	Widely Diverse
Documents	Text	HTML
No. of Matches	Small	Large
IR techniques	Content based	Link based

COMPONENTS OF A SEARCH ENGINE (Bruce Croft)

A search engine is the practical application of information retrieval techniques to large-scale text collections. Search engines come in a number of configurations that reflect the applications they are designed for. Web search engines, such as Google and Yahoo!, must be able to capture, or *crawl*, many terabytes of data, and then provide subsecond response times to millions of queries submitted every day from around the world.

Search engine **components support two major functions:**

1. **Indexing process**
2. **Query process**

1.Indexing process

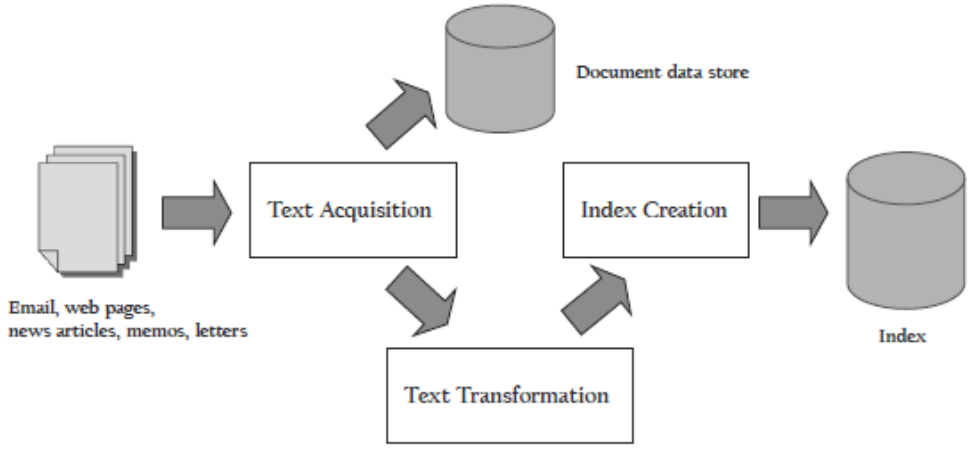


Fig. 2.1. The indexing process

The indexing process builds the structures that enable searching, and the query process uses those structures and a person's query to produce a ranked list of documents. Figure 2.1 shows the high-level "building blocks" of the indexing process.

These major components are

a) **Text acquisition**
b) **Text transformation**
c) **Index creation**

a)Text acquisition

The task of the text acquisition component is to identify and make available the documents that will be searched. Although in some cases this will involve simply using an existing collection, text acquisition will more often require building a collection by *crawling* or scanning the Web, a corporate intranet, a desktop, or other sources of information. In addition to passing documents to the next component in the indexing process, the text acquisition component creates a document data store, which contains the text and *metadata* for all the documents. Metadata is information about a document that is not part of the text content, such as the document type (e.g., email or web page), document structure, and other features, such as document length.

b)Text transformation

The text transformation component transforms documents into *index terms* or *features*. Index terms, as the name implies, are the parts of a document that are stored in the index and used in searching. The simplest index term is a word, but not every word may be used for searching. A "feature" is more often used in the field of machine learning to refer to a part of a text document that is used to represent its content, which also describes an index term. Examples of other types of index terms or features are phrases, names of people, dates, and links in a web page. Index terms are sometimes simply referred to as "terms." The set of all the terms that are indexed for a document collection is called the *index vocabulary*.

c)Index creation

The index creation component takes the output of the text transformation component and creates the indexes or data structures that enable fast searching. Given the large number of documents in many search applications, index creation must be efficient, both in terms of time and space. Indexes must also be able to be efficiently *updated* when new documents are acquired.

Inverted indexes, or sometimes *inverted files*, are by far the most common form of index used by search engines. An inverted index, very simply, contains a list for every index term of the documents that contain that index term. It is inverted in the sense of being the opposite of a document file that lists, for every document, the index terms they contain. There are many variations of inverted indexes, and the particular form of index used is one of the most important aspects of a search engine.

2.Query process

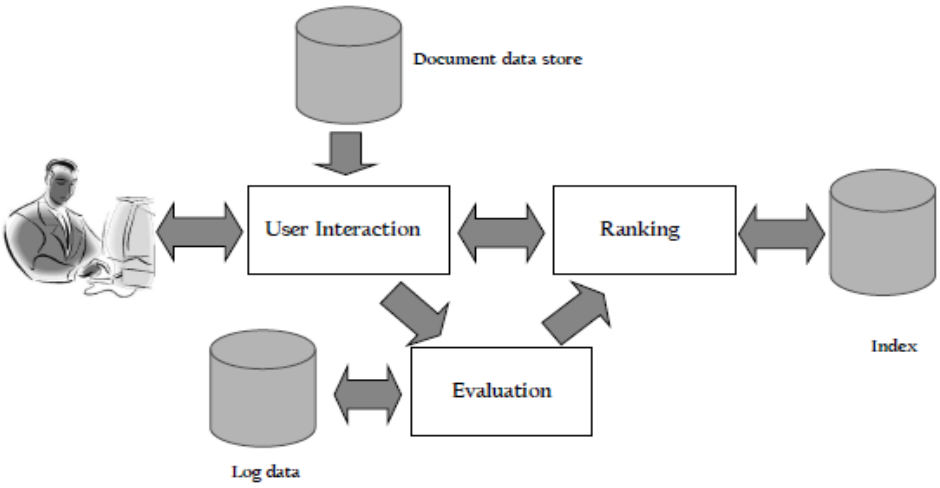


Fig. 2.2. The query process

Figure 2.2 shows the building blocks of the query process. The major components are

a) **User interaction**
b) **Ranking**
c) **Evaluation**

a)User interaction

The user interaction component provides the interface between the person doing the searching and the search engine. One task for this component is accepting the user's query and transforming it into index terms. Another task is to take the ranked list of documents from the search engine and organize it into the results shown to the user. This includes, for example, generating the *snippets* used to summarize documents. The document data store is one of the sources of information used in generating the results. Finally, this component also provides a range of techniques for refining the query so that it better represents the information need.

b)Ranking

The ranking component is the core of the search engine. It takes the transformed query from the user interaction component and generates a ranked list of documents using scores based on a retrieval model. Ranking must be both efficient, since many queries may need to be processed in a short time, and effective, since the quality of the ranking determines whether the search engine accomplishes the goal of finding relevant information. The efficiency of ranking depends on the indexes, and the effectiveness depends on the retrieval model.

c)Evaluation

The task of the evaluation component is to measure and monitor effectiveness and efficiency. An important part of that is to record and analyze user behavior using *log data*. The results of evaluation are used to tune and improve the ranking component. Most of the evaluation component is not part of the online search engine, apart from logging user and system data. Evaluation is primarily an offline activity, but it is a critical part of any search application.

CHARACTERIZING THE WEB (Ricardo Baeza)

Characteristics

- Measuring the Internet and in particular the Web, is a difficult task due to its highly dynamic nature.
    - There were more than 750 million computers in more than 200 countries directly connected to the Internet, many of them hosting Web servers.
    - Further, the estimated number of Web servers currently exceeds 206 million (Netcraft Web survey of June, 2010).
    - Considering these numbers, we can say that there is about one Web server per every four computers directly connected to the Internet.
- How many different institutions (not Web servers) maintain Web data?
    - This number is smaller than the number of servers, because many places have multiple servers.
    - The exact number is unknown, but should be larger than 40% of the number of Web servers.
- More recent studies on the size of search engines estimated that there were over 20 billion pages in 2005, and that the size of the static Web is roughly doubling every eight months.
- Nowadays, the Web is infinite for practical purposes, as we can generate an infinite number of dynamic pages (e.g. consider an on-line calendar).
- The most popular formats of Web documents are HTML, followed by GIF and JPG (both images), ASCII text, and PDF, in that order.
- The most popular compression tools used are GNU zip, Zip, and Compress.
- There are many characteristics and statistics of HTML pages, some of which are as follows.
    - First, most HTML pages are not standard, meaning that they do not comply with all the HTML specifications.
    - Second, HTML pages are small (around 10KB) and usually contain few images.

- The average page contains between 5 and 15 hyperlinks and most of them are local, that is, they point to pages in their own Web server hierarchy. On average, the number of external pages pointing to any given page, is close to zero!

- The most referenced sites are the main Internet companies.
- In the year 2000, it was estimated that around 70% of the pages were written in English, and that the numbers of words available in other languages was growing faster than the number of words in English. On January 2003, Google Zeitgeist showed that around 50% of the queries to Google were using English, down from around 60% in 2001.

Structure of the Web Graph

Web can be viewed as a graph in which the nodes represent individual pages and the edges represent links between pages. Broder et al. compared the topology of the Web graph to a bow-tie. This metaphor is schematically represented in Figure 11.1. With a bit of imagination, the reader can visualize a bow-tie where the largest strongly connected component (SCC) plays the role of the central knot of the tie.

Original "bow-tie" structure of the Web

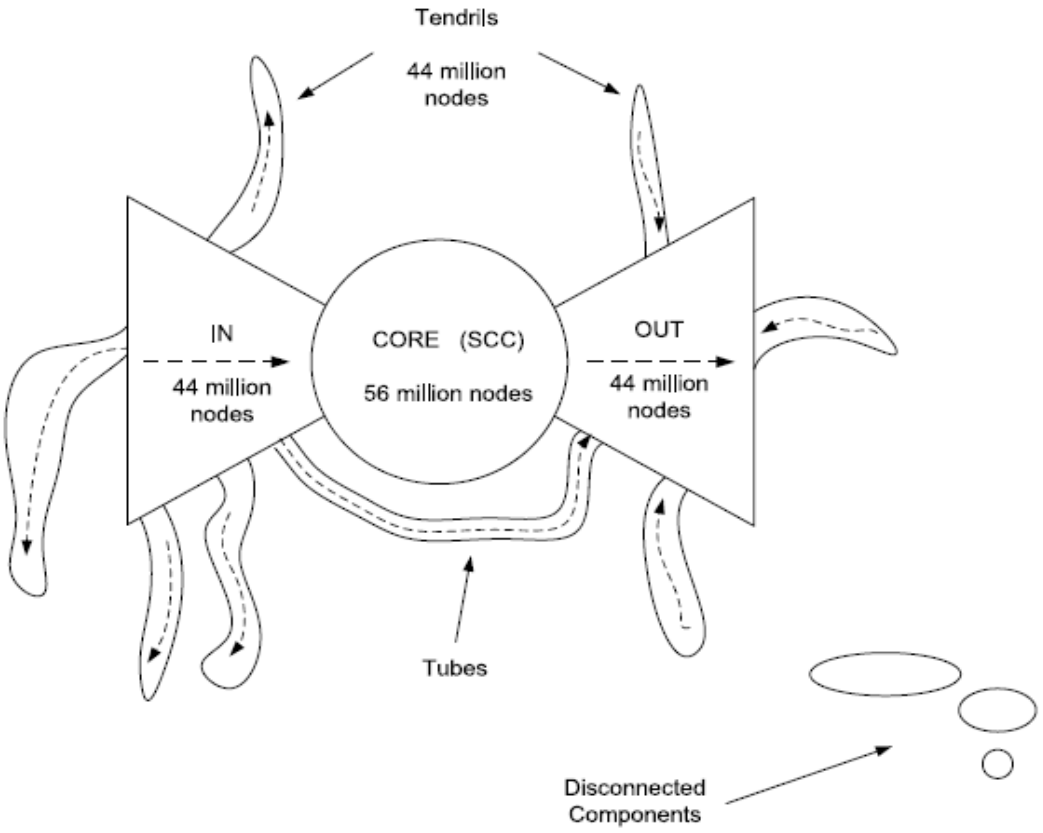


Figure 11.1: Original "bow-tie" structure of the Web (adapted from [271]).

Components in the Original "bow-tie" structure of the Web

(a) **CORE**: sites that compose the strongly connected component of the graph. By definition, one can navigate from any site in CORE to any other site in CORE.
(b) **IN**: sites that can reach sites in CORE but cannot be reached from sites in CORE.
(c) **OUT**: sites that can be reached from sites in CORE, but without a path to go back to CORE.
(d) **TUBES**: sites in paths that connect directly IN to OUT and that are outside CORE.
(e) **TENTACLES or TENDRILS**: sites that can be reached from sites in IN (T.IN) and sites that only reach sites in OUT (T.OUT), but that do not belong to the previous components.
(f) **DISCONNECTED or ISLANDS**: unconnected sites, whose connected component may have a structure similar to the whole Web.

Extended "bow-tie" structure of the Web

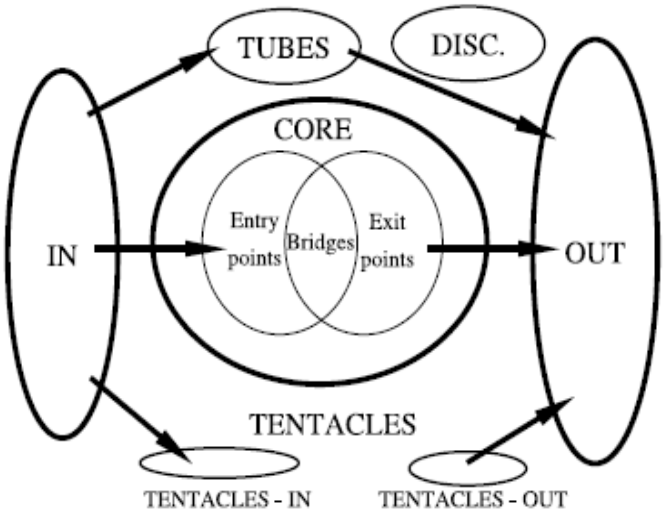


Figure 11.2: Schematic depiction of the macroscopic "bow-tie" structure of the Web.

Components in the Extended "bow-tie" structure of the Web

(a) **Bridges**: sites in CORE that can be reached directly from the IN component and that can reach directly the OUT component.

(b) **Entry points**: sites in CORE that can be reached directly from the IN component but are not in Bridges.

(c) **Exit points**: sites in CORE that reach the OUT component directly, but are not in Bridges.

(d) **Normal**: sites in CORE not belonging to the previously defined sub-components.

- **In all the (bounded) studies of the Web**, the CORE component is composed of a minority of the Web sites. On the other hand, it has a heavier density of Web pages.
- **From link analysis,**
  - There is a correlation between structure and quality of the content.
  - Some studies hint that the number of ISLANDS is much larger than we may think, as most of them are not connected to the Web and hence, not easy to find unless they are registered with the search engines.

Modeling the Web

1. Power law

All characterizations of the Web show that the quantitative properties of the CORE component follow a power law distribution. A general **power law** is a function that is invariant to scale changes. Its simplest form is:

$$f(x) = \frac{a}{x^{\alpha}} \quad \text{with} \quad \alpha > 0$$

Examples of Web measures that follow a power law include:

- the number of pages per Web site and the number of Web sites per domain
- the incoming and outgoing link distributions, as well as the number of connected components

This is true not only for the Web graph, but also for the host-graph, which is the connectivity graph at the level of Web sites.

Table 11.1 shows a summary of the main findings. For the page size, there are two exponents: one for pages with less than 20KB, and one for the rest. The same for the out-degree: one for pages with roughly less than 20 out-links, and one for pages with more out-links.

Region	Page Size		Pages	In-	Out-degree	
	Small	Large	per site	degree	Small	Large
Brazil	0.3	3.4	1.6	1.89	0.67	2.71
Chile	0.4	3.2	1.6	2.01	0.72	2.56
Greece	0.4	3.2	1.6	1.88	0.61	1.92
Indochina	n/a	n/a	1.2	1.63	0.66	2.62
Italy	n/a	n/a	1.3	1.76	0.68	2.52
South Korea	0.4	3.7	3.2	1.90	0.29	1.97
Spain	n/a	2.25	1.1	2.07	0.86	4.15
United Kingdom	n/a	n/a	1.3	1.77	0.65	3.61
World	n/a	n/a	n/a	2.1	n/a	2.7

Table 11.1: Summary of power-law exponents for the Web of various countries and regions.

2. Mathematical model

In addition, the distribution of document sizes can be also be described by a **mathematical model** that states that the document sizes are self-similar, that is, they are invariant to scale changes (a similar behavior appears in Web traffic). This best model is based in the mix of two different distributions. The main body of the distribution follows a **Logarithmic Normal distribution**, such that the probability of finding a document of size x bytes is given by

$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(\ln x - \mu)^2/2\sigma^2}$$

where the average (μ) and standard deviation are 9.357 and 1.318, respectively.

3. Heaps' and Zipf's laws

- These laws are also valid in the Web.
  - In particular, the vocabulary grows faster (larger *b*) and the word distribution should be more biased (larger *q*)
- **Heaps' Law**
  - An empirical rule which describes the vocabulary growth as a function of the text size.
  - It establishes that a text of *n* words has a vocabulary of size $O(n^b)$ for $0<b<1$
- **Zipf's Law**
  - An empirical rule that describes the frequency of the text words.
  - It states that the *i*-th most frequent word appears as many times as the most frequent one divided by i^q, for some $q>1$