Architecture of a search engine

Table of contents

A software architecture generally consists of:

- Software components
- Interfaces provided by those components
- Relationships between them

The two primary goals of a search engine are:

- Effectiveness (*quality*): To be able to retrieve the most relevant set of documents possible for a query

- Efficiency (*speed*): To process queries from users as quicky as possible Although not official, it is required as well that the system immediately react to changes in documents

Search engines employ **specialized data structures** that are optimized for **fast retrieval**.

Basic building blocks

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

- **Indexing process:** ***Builds the structures*** that enable searching
- **Query process:** ***Uses those structures*** and a person's query to produce a ranked list of documents.

Indexing process

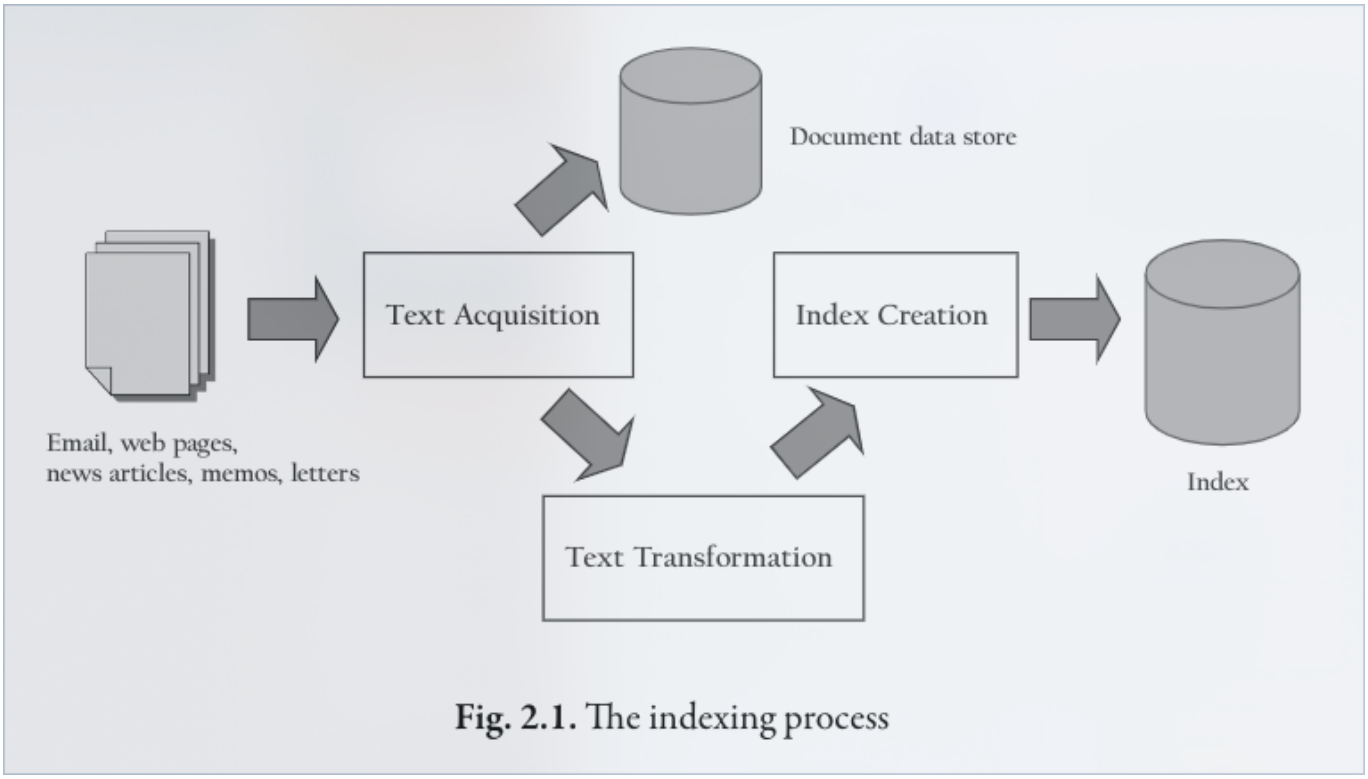


Fig. 2.1. The indexing process

The high-level building blocks of the indexing process are text acquisition, text transformation and index creation

Text acquisition:

- To identify and make available the documents that will be searched.
- Often, it will require building a collection by *crawling* or scanning the source of information.
- Creates a document data store, which contains for all documents:
  - Text
  - Metadata - Information about a document that is not part of the text content, but stores information about the document itself
    - Document type
    - Document structure
    - Document length

Crawler

Identifies and acquires documents for the search engine. A web crawler is designed to follow the links on the web pages to discover and download new pages.

- Site search crawler - are restricted to a single site
- Focused (topical) web crawler - use classification techniques to restrict the pages that are visited to those that are likely to be about a specific topic
- Enterprise document crawler - follows links to discover both external and internal (i.e., restricted to the corporate intranet) pages but scans other company information as well
- Document crawler - similar to the above, but in this case only the user’s personal directories need tobe scanned

Feeds

Document feedsare a mechanism for accessing a real-time stream of documents. A common standard is RSS

Conversion

Documents found by a crawler usually are in several formats and need to be converted to plain text for its successful analysis. It also requires to convert the documents to a consistent encoding scheme before further processing

Document data store

It is a database created to manage large numbers of documents and its structured data. It usually stores links and its anchor text. This saves the search engine the task of access and reprocess the original documents for each query

Text transformation:

- Transforms documents into:
    - *Index terms*: Parts of a document that are stored in the index and used in searching. The set of all the terms is called *index vocabulary*
    - *Feature*: Part of a text document that is used to represent its content, which also describes an index term

Parser

Processes the sequence of text tokens to recognize structural elements (titles, figures, links, headings)

Stopping

Removes common words from the stream of tokens that become index terms (the, of, to, for)

Stemming

Groups words that are derived from a common stem (fish, fishing, fishes -> fish)

Link extraction and analysis

Extraction means that the information (links, anchors) stored in the document data store can be indexed asides from the general text context. Whereas, analysis implies to provide the search engine with a rating of popularity (how important it is)

Information extraction

Used to identify terms that are more complex than single words

Classifier

Assigns predefined class labels to documents or parts of documents (sports, politics, business). Identifies spam and ads.

Index creation:

- Creates the **indexes or data structures that enable fast searching** from the output of the text transformation
- These indexes are supposed to be efficiently updated when new documents are acquired.
- The most popular form of index is called *inverted indexes*, they contain a list for every index term of the documents that contain that index term

Document statistics

Gathers and records statistical information about words, features and documents - counts index term ocurrences, positions where the index terms ocurred, concurrences over groups, etc.

Weighting

Reflects the relative importance of words in documents Uses *tf.idf* (term frequency.inverse document frequency)

Inversion

Core of the indexing process. Changes the stream of *document-term information* (from text transformation) -into-> *term-document information* (for the creation of inverted indexes)

Index distribution

Distributes indexes across multiple nodes for enable parallelism

Query process

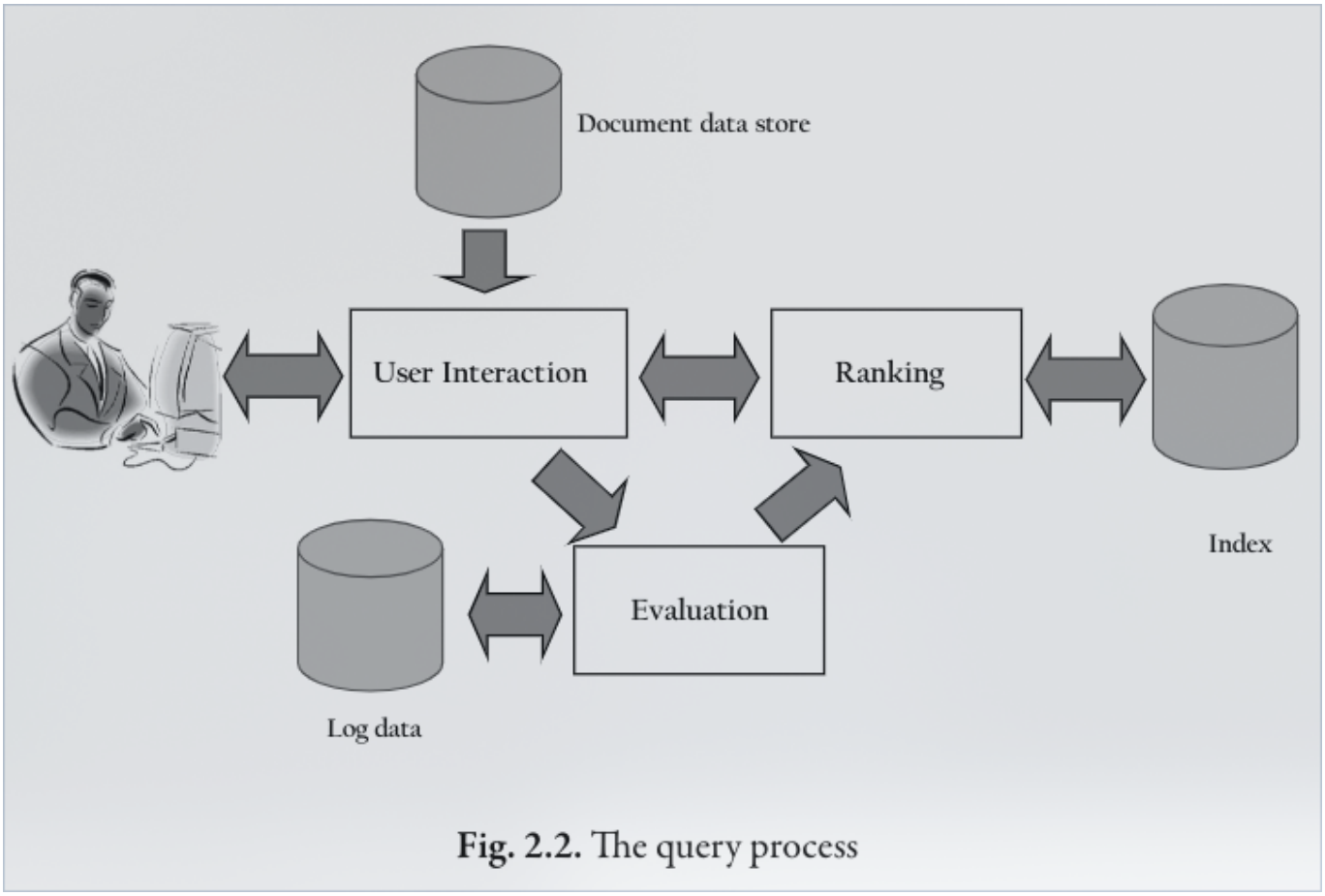


Fig. 2.2. The query process

The major-components of the query process are user interaction, ranking and evaluation

User interaction

- Provides the interface between the person doing the searching and the search engine
- Accepts the user's query and transforms it into index terms
- Takes the ranked list of documents from the search engine and organize it into the results shown to the user

Query input

Providex an interface and a parser for a *query language* (that contains a small number of operators that indiate the parser that the text should be treated in a special way)

Query transformation

Improves the initial query (before and after producing a document ranking). There are some techniques:

- Spell checking/query suggestion
- Query expansion
- Relevance feedback

Results output

Constructs the final display of ranked documents (from the ranking component), highlighting important words and adding relevant ads *(ew)*

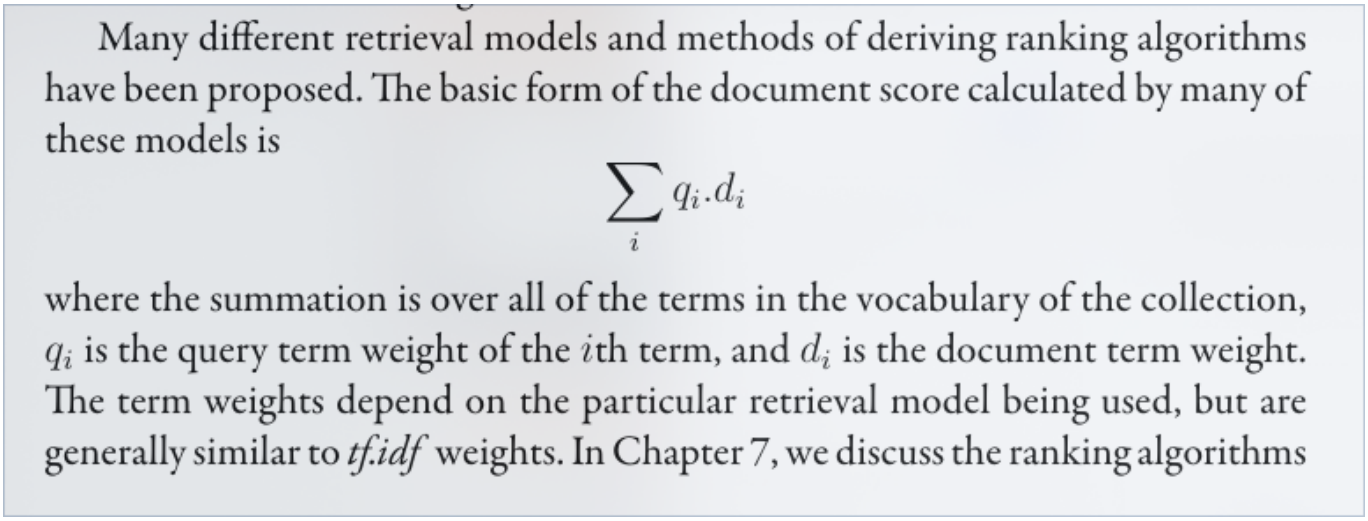
Ranking

Represents the core of the search engine

- Takes the transformed query from the user interaction component and generates a ranked list of documents using scores based on a retrieval model
- Its **efficiency depends on the indexes** and the **effectiveness depends on the retrieval model**

Scoring -query processing

Calculates scores for documents based on a retrieval model



Many different retrieval models and methods of deriving ranking algorithms have been proposed. The basic form of the document score calculated by many of these models is

$$\sum_i q_i.d_i$$

where the summation is over all of the terms in the vocabulary of the collection, q_i is the query term weight of the ith term, and d_i is the document term weight. The term weights depend on the particular retrieval model being used, but are generally similar to *tf.idf* weights. In Chapter 7, we discuss the ranking algorithms

Performance optimization

Involves the design of ranking algorithms and the as-sociated indexes to decrease response time and increase query throughput

Distribution

Basically, it means to allocate queries that potentially can be reused.

Evaluation

- Measures and monitors effectiveness and efficiency
- Records and analyzes user behavior using log data
- The results are used to tune and improve the ranking component