Chapter 2

Information Retrieval for Young Users

This chapter introduces the fundamental concepts and approaches that are prerequisites for the work described in this thesis. In Section 2.1 an introduction to Information Retrieval (IR) is given by defining the research field and describing IR system architecture. Furthermore, this section provides explanation of relevance ranking and introduction to search user interfaces. An introduction to targeted search engines is given in Section 2.1.4. The main aspects of children's development relevant to IR tasks are described in Section 2.2. In Section 2.3, basic user research methods and user evaluation types are briefly discussed.

2.1 Basics of Information Retrieval

The research field of Information Retrieval was defined by Salton [167, p. v] as:

Definition

"*Information retrieval* is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information."

Information retrieval is an activity that a user is engaged in. The user has a perceived gap in his or her knowledge, also called *information need*. This information is assumed to be present in a collection. In order to find this information, the user interacts with an IR system. The most common scenario is a search in a collection of text documents. In this thesis, we focus on textual information retrieval that is important in order to make the search in a web document collection work. Multimedia retrieval that deals with visual and sound data, e.g. images, music, videos [10, Chapter 14], is out of the scope of this thesis.

T. Gossen, *Search Engines for Children,*
DOI 10.1007/978-3-658-12069-6_2, © Springer Fachmedien Wiesbaden 2015

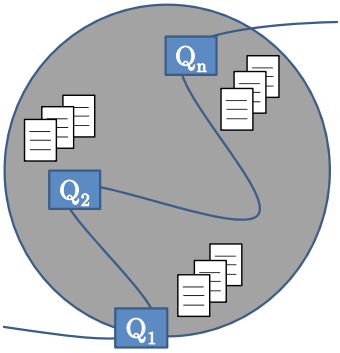

Fig. 2.1: Berrypicking search consists of a series of user interaction with an IR system. User learn bits of information one step at a time. Starting with a query Q_1, the user learns new information from the retrieved documents. This may lead to new queries until the user's information need is satisfied (this figure is modified from [13]).

The human interaction process with an IR system may be complex [13, 128]. Starting with one information need that may be vague in nature, a user submits a query and views the search results. They learn new information from these results, such as new facts or new vocabulary, that might lead to a drift in their information need or the understanding that rephrasing the query will lead to better search results. Therefore, they continue to query the search engine until the information need is satisfied. In other words, not every information need can be resolved with a single query and a single set of search results, but a series of queries and user examining the results might be necessary where a user is learning bits of information one step at a time. This bit-at-a-time retrieval is called berrypicking [13] by the analogy of picking berries from a bush (see Fig. 2.1).

Manning et al. [125, p. 1] defines Information Retrieval pointing out the unstructured nature of documents:

Definition

"*Information retrieval (IR)* is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)."

This means that in IR we deal with data which does not have a clear semantic structure. This leads to differences between data and information retrieval. The main differences are summarized by van Rijsbergen [164] as in Table 2.1. In data retrieval our goal is to find an exact match that is conformant with our query. An example of data retrieval is searching in a relational database [32]. A database has an underlying schema. Therefore, it is possible to find the exact database entries that match the query. In information retrieval we are interested in best matches, even if these matches are only partial. In some cases the answer to a user's information need is spread across several documents and these two partial matches reveal the desired information (see Berrypicking model above – Fig. 2.1). The query language for data retrieval is usually artificial with a restricted syntax and vocabulary. This language allows a user to give an exact and complete specification of what is wanted. In IR an exact specification is not possible because of the unstructured nature of text documents. To specify their information need, users in IR mainly use the vocabulary of natural languages. The retrieved results in IR indicate the likelihood of their relevance to user's information need and are usually sorted according to the relevance. Relevance is a measure of how closely a given document matches a user's information need. This judgment is done by the user and depends on different factors, e.g. his domain knowledge, the context of the search or previously seen results. As we deal with probabilities, in IR small errors in matching generally are not critical, whereas errors in data retrieval imply a total failure of the system.

2.1.1 Architecture of an IR System

Fig. 2.2 shows the main components of an IR system according to Baeza-Yates and Ribeiro-Neto [10]. An IR system has a frontend, also known as a

	Data Retrieval	Information Retrieval
Matching	Exact match	Partial match, best match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic
Classification	Monothetic	Polythetic
Query language	Artificial	Natural
Query specification	Complete	Incomplete
Items wanted	Matching	Relevant
Error response	Sensitive	Insensitive

Table 2.1: Difference between data and information retrieval [164].

search user interface (SUI). The user has to formulate their information need in a specific form that can be understood by the IR system. Most common search engines allow to input a textual search query. Using the SUI the user submits a search query and also receives the visualised search results for the query.

The backend of an IR system manages the retrieval of relevant results. It requires a document collection which usually consists of text documents. In contrast to a document collection that is already in place, e.g. stored on the hard drive of a computer, web documents are scattered on the Web and located on different servers. Therefore, a web search engine has to gather web pages as a first step. This procedure is done by a crawler. The crawler collects information in a central location to be further analysed.

Web documents are usually in a HTML format that uses markup tags, e.g. indicating the body or font size. Therefore, web documents are parsed to extract the actual content. However, this content is only a sequence of characters and has to be processed in order to detect a well-defined sequence of linguistically-meaningful units [149]. This procedure is called pre-processing and consists of several steps. The main steps are tokenization, stopword elimination and stemming.

The first step is to convert the character sequence into a set of meaningful information terms (tokens), that are words or phrases (in case of names such as San Francisco) [126, p. 124]. Some of these tokens, e.g. articles and prepositions, are extremely common words which are of little value in helping to select documents matching the user's query. These tokens are called stop words and can be omitted [126, Chapter 15]. However, a user query may only contain stop words, e.g. "to be or not to be" (William

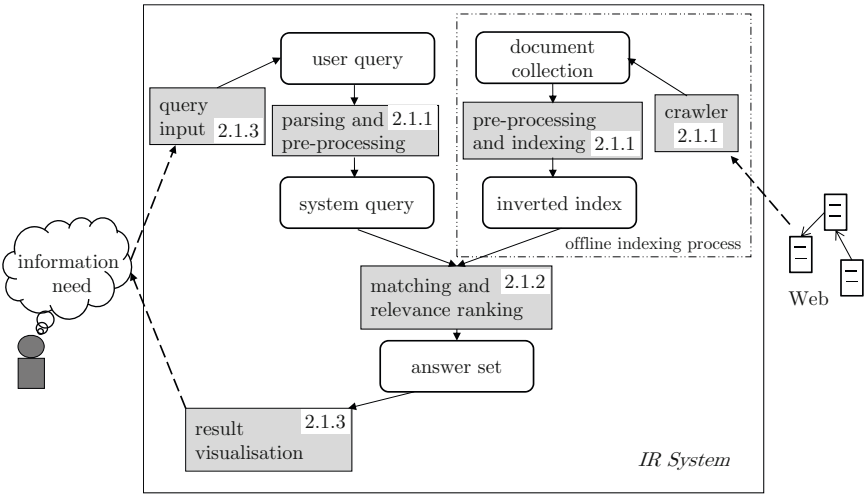

Fig. 2.2: Information flow in the high-level software architecture of an IR system based on [10]. Numbers indicate the section where the process is discussed in more detail.

Shakespeare, Hamlet). Removal of stop words makes it impossible to answer such queries. For this reason, many modern search engines do not remove stop words but rather assign a lower weight during the relevance calculation (see Section 2.1.2).

The next step of pre-processing is stemming where related forms of a word are reduced to a common base form (stem) [126, Chapter 15]. For grammatical reasons, documents contain different forms of a word, such as “process”, “processing”, “processed”. Stemming make it possible, given a search term in one form, to find the documents that also contain a search term in other grammatical forms. Stemming algorithms are language dependent. The widely used stemming algorithm for English language is Porter stemmer that performs rules based suffix removal [158].

The search query has to be parsed and pre-processed as well. Parsing is done in order to detect special query types, e.g. phrase queries (exact occurrence of a series of words) that are defined by quotation marks.

The output of document pre-processing are terms that will be indexed, i.e. stored in the internal data storage. The data storage contains references

from each term to the documents that contain this term (see Fig. 2.3). For this, IR systems use a data structure called inverted index [195]. The inverted index maps terms back to the parts of a document where they occur. The inverted index has a dictionary with indexed terms and corresponding postings lists. Each term in the dictionary refers to a posting list, i.e. a sorted list of document, usually represented by document IDs, containing the term. There are several extensions of an inverted index like positional index for proximity queries, biword index for phrase queries, q-gram index [137] and permuterm index [67] for wild-card queries. These specific indexes store additional information to enable different types of queries, e.g. storing term position within a document makes it possible to answer phrase queries.

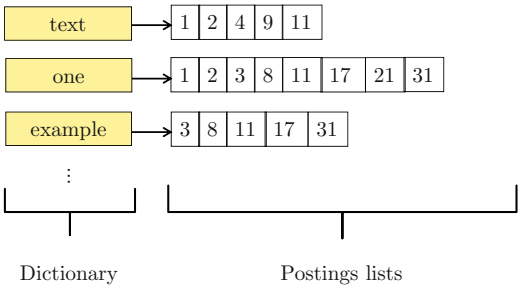

Fig. 2.3: Example of an inverted index based on [125]. The inverted index consists of the dictionary with index terms and the postings lists with document IDs that contain the corresponding terms. According to the presented inverted index, the term "text" can be found in documents $1, 2, 4, 9, 11$.

After query pre-processing, a retrieval model is used to match search query and data collection, and to conduct relevance ranking. There exist different models, e.g. boolean, vector space, probabilistic and language model (see [10, 125]).

The process of crawling, document pre-processing and indexing is performed offline, while a user's interaction with a system and relevance ranking of documents to the user's query are done in real-time. In the following, the most commonly used retrieval approaches, vector space model,

and the link-based algorithms for web retrieval, PageRank and HITS, are described.

2.1.2 Relevance Ranking

The goal of relevance ranking is to estimate the relevance of a document to a user query. Documents are then ranked according to the likelihood of relevance to the user. Here we describe several retrieval approaches that are largely used. We start with the vector space model. Then, two link-based models, PageRank and HITS, are introduced.

Vector Space Model: The vector space model [10, 169] considers each document d within the document collection D to be a vector in a vector space. The dimensions of the vector space are the index terms. The index terms may be weighted according to their importance for the corresponding document, or be binary (1 if the term is present in the document and 0 if not). An illustration of the vector space model based on a three-dimensional example is given in Fig. 2.4. In general, each document is a t-dimensional vector:

$$d_j = (w_{1,j}, w_{2,j}, ..., w_{t,j}), \tag{2.1}$$

where $w_{i,j}$ represents the weight of the ith term for this document. A query is represented as a t-dimensional vector as well:

$$q = (w_{1,q}, w_{2,q}, ..., w_{t,q}), \tag{2.2}$$

where $w_{i,q}$ represents the weight of the ith term for this query.

The most popular weighting approach combines the term frequency (TF) and the inverse document frequency (IDF) [170]. TF counts the number of appearances of the term in the document. IDF takes into account the frequency of the term in the collection:

$$IDF_i = log\frac{N}{df_i}, \tag{2.3}$$

where N is the total number of documents in the collection and df_i is the number of documents the term i occurs in. Then the calculation of the weight for term i and document j using TFIDF is as follows:

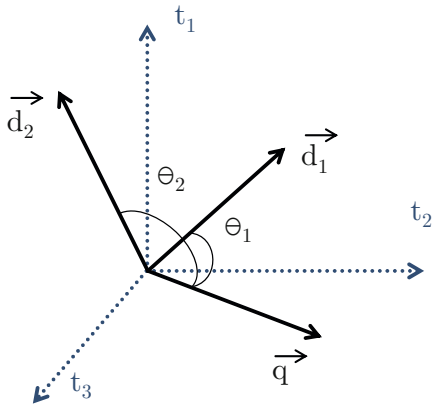

Fig. 2.4: Illustration of the vector space model based on [169]. Three-dimensional example with two documents d_1, d_2 and a query q is shown. The query is treated as a document in the space as well. Given the query, the documents are ranked according to their similarity to the query, for instance, by the cosine of the angle between the document and the query vector. The document ranking is d_1, d_2 as $\theta_1 < \theta_2$.

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{2.4}$$

Thus a term has a high weight in case this term is frequent in the document but rare in the document collection. Frequent words that occur in all the documents (stop words) receive a term weight equal to zero.

The vector model measures the relevance of a document given a query by the degree of similarity between the corresponding two vectors. A common similarity measure is the cosine similarity that calculates the cosine of the angle between the two vectors:

$$sim(d_j, q) = cos(\theta) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{|\mathbf{d_j}| \times |\mathbf{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sum_{i=1}^{t} w_{i,j}^2 \times \sum_{i=1}^{t} w_{i,q}^2} \tag{2.5}$$

The cosine similarity has a built-in document length normalization.

The vector space model has several important advantages, e.g. over the Boolean model[1]. Its ranking scheme improves retrieval quality. It allows matching of documents that may contain only some of the query terms. The vector space model supports the ranking of documents and sorting the results according to their similarity to the query. In theory, the vector space model has some limitations. In the vector space model, documents and queries are represented by their terms, as a bag of terms. The order of terms is ignored. Furthermore, the assumption is made that the terms are all mutually independent. However, it is not true for natural languages. Despite of the limitations, the model is a reliable ranking method for general document collections.

PageRank: The PageRank algorithm [23, 147] is a link-based approach for web retrieval which was part of the ranking algorithm originally used by the Google web search engine. It exploits the link structure of the web. The web can be seen as a directed graph with web pages as nodes and hyperlinks between them as edges (see Figure 2.5). PageRank determines the importance or quality of a particular page using the information obtained from the link structure. Each page has a rank. An intuitive assumption is made that a page has a high rank value if it is linked to by many other pages with high rank values.

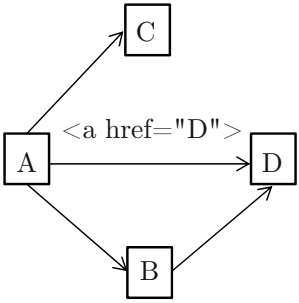

Fig. 2.5: Illustration of a web graph. Four pages A,B,C and D are graph nodes. The graph is directed. For example, hyperlinks of page A result in graph edges to B,C and D. Page D does not contain any hyperlinks.

[1] The boolean model only considers, if a query term is contained in a document. The query may contain boolean operators (AND, OR, NOT), which are transformed into set operations (intersection, union, set negation) on the result sets for individual terms [10].

Let u be a web page. Then the PageRank $R(u)$ of u is calculated as follows:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}, \tag{2.6}$$

where B_u is the set of pages that have a link to u. N_v is the number of outgoing links from v. c is normalization factor. Thus, the rank of a page is divided among its outgoing links equally to contribute to the ranks of the pages they point to. This simulates a random web surfer staying on one web page with N_v outgoing links. This surfer can follow one of the links with $\frac{1}{N_v}$ probability.

However, the formula 2.6 does not work in case the web graph has dead ends (pages with no outgoing links) and cycles. To prevent this, an additional parameter, damping factor d, is introduced that simulates the ability of a web surfer to teleport from one page to another using other methods than following links, e.g. given the URL-address directly in the browser.

$$PR(u) = (1 - d) + d \sum_{v \in B_u} \frac{R(v)}{N_v}, \tag{2.7}$$

The PageRank can be calculated using a simple iterative algorithm. It corresponds to the principal eigenvector of the normalised link matrix of the web.

PageRank is query independent. Therefore, in web retrieval PageRank is combined with other models, e.g. with the vector space model [10]. Just to give an idea, a very simple approach would be a linear combination of the two measures.

HITS: Another link-based ranking approach is called Hyperlink-Induced Topic Search (HITS) and was developed by Kleinberg [105]. The HITS algorithm operates with a set of pages that are the most relevant pages to a user query (root set). This set can be, for example, obtained using the vector space approach. Then, the root set is expanded to a base set by including the web pages that are linked from it and the pages that link to it. The pages in the base set and the hyperlinks between them form a focused subgraph where all the calculations are performed.

HITS distinguishes between pages that are an "authoritative" information source on a certain topic and "hub pages" that are link collection pages referring to authoritative pages. Hubs and authorities have a mutually reinforcing relationship, i.e. a good hub points to many good authorities and

http://www.springer.com/978-3-658-12068-9

Search Engines for Children
Search User Interfaces and Information-Seeking Behaviour
Gossen, T.
2015, XXIII, 283 p. 62 illus., Softcover
ISBN: 978-3-658-12068-9