Document Information Retrieval

Chapter · March 2007

DOI: 10.1007/978-1-84628-726-8_16

CITATIONS
5

READS
1,847

5 authors, including:

Stefan Klink
Duale Hochschule Baden-Württemberg Karlsruhe
44 PUBLICATIONS **577** CITATIONS
SEE PROFILE

Koichi Kise
Osaka Prefecture University
331 PUBLICATIONS **3,190** CITATIONS
SEE PROFILE

Andreas Dengel
Deutsches Forschungszentrum für Künstliche Intelligenz
931 PUBLICATIONS **13,281** CITATIONS
SEE PROFILE

Markus Junker
Deutsches Forschungszentrum für Künstliche Intelligenz
37 PUBLICATIONS **578** CITATIONS
SEE PROFILE

16

Document Information Retrieval

Stefan Klink, Koichi Kise, Andreas Dengel, Markus Junker,
and Stefan Agne

16.1 Introduction

Due to the widespread use of ubiquitous electronic tools like laptops, PDAs and digital cameras for the acquisition, production and archiving of documents, more and more information exists in electronic form. The ease with which documents are produced and shared in the World Wide Web has led to a potentiation of information reachable by each user. This has created a growing demand of adaptive and intelligent access to relevant information.

A study of the International Data Corporation (IDS) shows that the capacity of data in enterprise networks will increase from 3200 petabytes in the year 2002 up to 54,000 petabytes and growing in the year 2004. Cleverdon estimates that the number of new publications in the most important scientific journals will be 400,000 per year [1]. Storing this mass of information is a problem, but searching specific information is a challenge that has become an even greater object of public concern. These tremendous masses of data make it very difficult for a user to find the "needle in the haystack" and nearly impossible to find and flip through all relevant documents and images. Because a human can no longer gain an overview over all information, the risk of missing important data or of getting lost in the haystack is very high.

The task of document information retrieval is to retrieve relevant documents in response to a query that describes a user's information need. Its basic operation is to measure the similarity between a document and a query. Documents with high similarity are presented to the user as the result of retrieval. Although this research field has several decades of history, there still exist some open problems.

The rest of this chapter is organized as follows. In Section 16.2 we give an overview of the vector-space model and basic techniques commonly

utilized in document retrieval. In Section 16.3 three innovative applications are introduced that make use of these techniques.

16.2 Document Retrieval Based on the Vector-Space Model

Throughout the previous few decades of information retrieval science, many models have been invented and a huge amount of variations have been tested. In the field of document retrieval, only a few are established. The most popular retrieval model is the vector-space model (VSM) introduced by Salton [2–4].

16.2.1 Identification of Index Terms

The fundamental unit of an automatic statistical indexing is a word. Each word of a document can be seen as a discriminative feature between the current document and all other documents in the document collection. This feature can be quantified and could also be negative.

During the late 1950s, Luhn showed that the most discriminative words are those that occur with a relatively average frequency within a document collection [5]. If a word – e.g. a pronoun or preposition – occurs very often, then it cannot characterize the content of a document. On the other hand, words occurring very rarely in documents are also rarely used in a user's query. These observations are the foundation of frequency-based techniques for automatic term extraction methods.

Nowadays, it is common not to extract single words according to their frequency but to simply extract all words of the documents and then assign them a frequency-based weight. Words with high frequency will get weights near zero. Due to memory restrictions, they will probably not be included in the index, or they will be skipped over entirely in the case that a *stop word list* is employed. Such a list contains functional words like "with", "also", "can", "the", etc. Furthermore, this list may contain words with no discriminative meaning or words that are given by the semantic of the document collection – e.g. 'computer' or 'program' in a collection of computer science documents.

Another linguistic problem is that the same meaning of a word can be represented by several similar but not identical words – e.g. "sofa" vs. "couch". In the same manner, singular and plural of nouns are often different. This is a significant problem for matching functions that use just the exact occurrence of query terms within the documents.

Several techniques have been proposed to avoid this problem. They are mostly based on a mapping of words to a set of descriptors for word families. Such methods that identify variants of word forms are also known as

conflation methods. A widely used morphological approach is *word stemming*, also known as *suffix stripping*. It is based on a rule-based removal of derivative and declined word suffixes and reduces each word to its stem, which is not necessarily a complete word as used in a document. Of course, a list of word endings is not able to follow a set of grammatical rules. Therefore, it is not possible to implement a system with 100% accuracy. But for some languages, such as English or German, an algorithm developed by Porter is a suitable method for word stemming [6–8].

Approaches to identify variants of word forms are not just based on cutting off the suffix. A more general approach for combining words makes use of the comparison of two words based on the number of *n-grams*, which means sub-strings with n characters occurring in both words. This approach is able to identify words as related independent to a certain prefix or suffix [9].

Lewis and Spärck Jones showed that the removal of stop words and word stemming is mainly used to reach two problematical aims: discrimination and normalization [10]. On the one hand, documents must be discriminative to be able to find a specific document out of the collection. On the other hand, documents must be normalized so that a comparison of the user query with the documents is successful and query term are compared on the concept level and not on the exact character level.

The selection of algorithms for word stemming is one of the factors that influence the trade-off between the two aspects above. Indexing will be always just sub-optimal. Kupiec elucidates this fact with a reference to the book and film title *Gone With the Wind* [11]. With standard indexing algorithms, this will be reduced to "go wind". Another example that shows the problem even better is the word "trainers" (sport shoes), which will be reduced to "train" (railway).

The effects of word stemming are fundamental and pervade the complete retrieval process from the beginning (indexing) up to the satisfaction of the user need (query result). The problems shown here occur all over and are often discussed. But over-normalization and sub-optimal discrimination are just two main problems during automatic indexing. Further approaches tackle the problem of how to handle spelling, inappropriate hyphenation and the occurrence of alphanumerical words such as "2005".

In general during the balancing act of indexing, the profit from normalization is higher than the loss from discrimination methods.

16.2.2 The Simple Vector-Space Model

In this section we explain the vector-space model, which is the basis of our work. Essential parts of this model are the representation of documents and queries, a scheme for weighting terms and an appropriate metric for calculating the similarity between a query and a document.

Representation of Documents and Queries

The task of traditional document retrieval is to retrieve documents that are relevant to a given query from a fixed set of documents – i.e. a document database. A common way to deal with documents, as well as queries, is to represent them using a set of index terms (simply called *terms*) and ignore their positions in documents and queries. Terms are determined based on words of documents in the database. In the following, t_i $(1 \leq i \leq m)$ and d_j $(1 \leq j \leq n)$ represent a term and a document in the database, respectively, where m is the number of terms and n is the number of documents. In the VSM, a document is represented as an m-dimensional vector

$$\boldsymbol{d}_j = (w_{ij}, \ldots, w_{mj})$$

Similarity Measurements

The result of the retrieval is represented as a list of documents ranked according to their similarity to the query. The selection of a similarity function is a further central problem having decisive effects on the performance of an IR system. A detailed overview of similarity functions can be found in [20].

A common similarity function in text-based IR systems is the cosine metric, which is also used in the SMART system and in our approach. For this metric, the similarity $sim(\boldsymbol{d}_j, \boldsymbol{q})$ between a document $\boldsymbol{d}_j$ and a query $\boldsymbol{q}$ is measured by the standard cosine of the angle between the document vector $\boldsymbol{d}_j$ and the query vector $\boldsymbol{q}$:

$$sim(\boldsymbol{d}_j, \boldsymbol{q}) = \frac{\boldsymbol{d}_j^T \cdot \boldsymbol{q}}{\| \boldsymbol{d} \| \cdot \| \boldsymbol{q} \|}, \tag{16.4}$$

where $\| * \|$ is the Euclidean norm of a vector.

16.2.3 Relevance Feedback

Because the retrieval techniques discussed in this work belong to the family of relevance feedback techniques, this section gives a short overview of relevance feedback techniques first and our approaches are introduced later.

Taxonomy of Relevance Feedback Techniques

A common way of searching information is to start with a short initial query and to reformulate it again and again until satisfying results are returned. To do this, a lot of effort and time has to be invested by the user.

The main idea of relevance feedback techniques is to ask the user to provide evaluations or *relevance feedback* on the documents retrieved from the query. This feedback is then used for subsequently improving the retrieval effectiveness in order to shorten the way to more satisfying results.

The feedback is given by marking just the relevant documents in the result list or more specifically by marking the relevant and the irrelevant documents. The marking itself can be boolean (marked or not) or within a given scale in more advanced systems.

In general, relevance feedback techniques are not restricted to specific retrieval models and can be utilized without a document assessment function that is responsible for the ranking of the retrieved documents.

In literature, many different strategies are described and many implementations have been tested. Figure 16.1 shows how the two main strategies for relevance information are utilized: (1) the user's search query is reformulated, and (2) documents of the collection are changed.

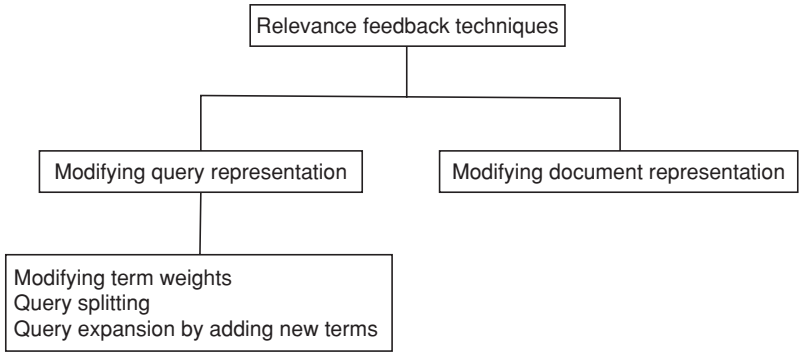

Fig. 16.1. A taxonomy of relevance feedback techniques.

Both approaches have their pros and cons. Approaches that reformulate the search query only influence the current search and do not affect further queries from the same or other users. On the other hand, approaches that change the documents within the collection possibly do not affect the current search.

Especially of interest is the way in which the query is reformulated because the approach described in this work also tries to answer this question.

In general, all introduced approaches in the following sections reach improvements after some number of iterations.

Changing the Document Collection

Information retrieval methods that change documents within the collection are also known as *document transformation methods* [21] or *user-oriented clustering* [22, 23].

The hope of information retrieval in the vector-space model lies in the fact that the vector of a document relevant to the user's query is located near to that of the query. Document transformation (DT) approaches aim to improve those cases where this is not accomplished by moving the document vectors relevant to the query to the direction of the query. Those vectors that are irrelevant are moved further away.

When moving the document vectors (either closer to or away from the query vector), close attention must be paid so that each single movement is very small. The main reason for this is that the assessment of a particular user is not necessarily in agreement with those of other users.

DT methods were first described in the late 1960s by the SMART team [12, 24]. It is one of the strategies that is easy and efficient enough to be part of big search engines. Although these approaches were introduced early on, they achieved little attention and were only tested in a restricted way. Twenty years later, Salton identifies the main reason for this negligence [25, p. 326]:

Document-space modification methods are difficult to evaluate in the laboratory, where no users are available to dynamically control the space modifications by submitting queries.

In literature, several algorithms for DT can be found. Some examples are given in [21]:

$$D = (1-\alpha)D + \alpha\frac{\parallel D \parallel}{\parallel Q \parallel}Q \tag{16.5}$$

$$D = D + \beta Q \tag{16.6}$$

$$D = D_{original} + D_{learned}, \quad \text{whereas} \tag{16.7}$$

$$D_{learned} = \begin{cases} D_{learned} + \beta\ Q & \text{if } \parallel D_{learned} \parallel < l; \\ (1-\alpha)\ D_{learned} + \alpha\frac{\|D_{learned}\|}{\|Q\|}\ Q & \text{otherwise.} \end{cases}$$

Here Q is the user query, D is the relevant document and l is a threshold.

The strategy in (16.5) ensures that the length of the document vector stays constant. Brauen [12] and Savoy et al. [26] have shown that this strategy is able to improve the retrieval results on small- and mid-sized document collections.

The strategy in (16.6) is the most simple, but it performs well on a low quantity (less than 2000) of queries [21]: terms of Q are weighted with β and then added to the document vector D. It should be noted that this method of summation causes the length of the document to grow without limitation.

Both strategies are sensitive to a supersaturation. If many queries are assigned to a relevant document, then the effect of the initial document terms is decreased with the growing amount of queries and the effect of the query terms dominates. This document saturation is a serious problem in search engines that utilize variants of these formulae.

The strategy in (16.7) was developed to solve this problem. With a growing number of queries (document length), the effect of queries is decreasing.

In general, DT techniques have been shown to improve retrieval performance over small- and medium-sized collections [12, 26]. There was no winner among the strategies that have been tried; different strategies perform best on different collections [26].

One notable and important difference between the DT methods and the methods modifying the query described next is that only the first ones leave permanent effects on a system.

Reformulation of the Query

As opposed to DT methods that try to move relevant documents nearer to their appropriate queries, methods for reformulating the query try to

solve the retrieval problem from the other side. They try to reformulate the initial user query in a way that the query moves nearer to the relevant documents.

Three basic approaches to improve the retrieval results are known in literature: (1) methods that modify weights of the query terms, (2) methods for query splitting and (3) methods for query expansion by adding new terms. The approach in Section 16.3.1 used in our work belongs to the latter group, which is most important.

Modification of Term Weights

Methods for modifying weights of query terms do not add any terms to the initial query but merely increase or decrease the available weights with the help of the feedback information. The main problem of this approach is that no additional information (i.e. new terms) is placed in the query.

Query Splitting

In some cases, relevance feedback techniques supply only unsatisfying results – e.g. documents marked as relevant are not homogeneous, meaning that they do not have a single topic in common and do not form a common cluster in the vector space. Another problem is irrelevant documents that lie near (or in between) relevant documents. In this case, the initial query vector will be moved also to these irrelevant documents by the feedback.

To discover these cases, a common method is to cluster the documents marked as relevant and therewith to analyse if the documents share a topic and if they are homogeneous in the vector space. If the relevant documents are separable into several distinct clusters, then the initial query is split appropriately into the same number of sub-queries. The term weights are adjusted according to the document clusters [23].

Query Expansion

The third and in general most distributed group of methods for modifying the user query is the expansion of the query by adding new terms. These new terms are directly chosen with the help of user feedback after the presentation of the retrieved document. The selected terms (documents) are added to the initial query with appropriate weights.

Experimental results have shown that positive feedback – i.e. marking only relevant documents – is better than using positive and negative feedback in general. The reason is that documents within the relevant set are positioned more homogeneously in the vector space than those in the negative set.

Rocchio Relevance Feedback

Rocchio [27] suggested a method for relevance feedback that uses average vectors (centroids) for each set of relevant and irrelevant documents. The

new query is formed as a weighted sum of the initial query and the centroid vectors. Formally, the Rocchio relevance feedback is defined as follows:

Let q be the initial query, n_1 the number of relevant documents and n_2 the number of irrelevant documents. Then the new query q' is formed by:

$$\boldsymbol{q}' = \boldsymbol{q} \;+\; \frac{1}{n_1} \sum_{relevant} \frac{\boldsymbol{D}_i}{\parallel D_i \parallel} \;-\; \frac{1}{n_2} \sum_{non-relevant} \frac{D_i}{\parallel D_i \parallel}. \quad (16.8)$$

An important characteristic of this method is that new terms are added to the initial query and the former term weights are adjusted. Salton and Buckley [28] have tested a mass of variants of this linear vector modification. They asserted that this technique needs only a low calculation effort and in general it achieves good results. But they also observed that the performance varies over different document collections. Furthermore, they stated that these techniques have bigger gains in cases with poor initial queries compared to cases where the initial query provides very good results.

Pseudo-Relevance Feedback

The Rocchio relevance feedback of the previous section supplies good results but it has a crucial disadvantage: it needs user feedback. This is very hard to get in real IR systems because few users are willing to do the job of assessing documents.

One idea to simulate this explicit user feedback is to rely on the performance of the IR system and to assume that the top n_1 documents in the ranked document list are relevant. These are used as positive feedback for the relevance feedback method.

In contrast to the Rocchio relevance feedback, no negative feedback is considered. It may be possible to assume that the bottom n_2 documents are irrelevant and use them as negative feedback. But this variation is uncommon and generally leads to poorer results.

Like the Rocchio relevance feedback, the pseudo-relevance feedback works in three steps: (1) The initial query is given to the system and the relevant documents are determined. (2) In contrast to Rocchio relevance feedback, these relevant documents are not presented to the user for marking, but rather the n most similar documents are selected automatically to reformulate the query by adding all (or just certain) terms of these documents to the query. (3) The reformulated query is given to the system and the relevant documents are presented to the user.

An interesting variation of the pseudo-relevance feedback is described in Kise et al. [29]: Let $\boldsymbol{E}$ be a set of relevant document vectors for expansion given by

$$\boldsymbol{E} = \left\{ \boldsymbol{d}_j^+ \middle| \frac{sim(\boldsymbol{d}_j^+, \boldsymbol{q})}{\max_{1 \le i \le N} sim(\boldsymbol{d}_i^+, \boldsymbol{q})\}} \ge \theta \right\}, \quad (16.9)$$

where $\boldsymbol{q}$ is the original query vector, $\boldsymbol{d}_j^+$ a document vector relevant to the query and θ a similarity threshold. The sum $\boldsymbol{d}_s$ of these relevant document vectors,

$$\boldsymbol{d}_s = \sum_{\boldsymbol{d}_j^+ \in \boldsymbol{E}} \boldsymbol{d}_j^+, \tag{16.10}$$

can be considered as enriched information about the original query.[1] With this, the expanded query vector $\boldsymbol{q}'$ is obtained by

$$\boldsymbol{q}' = \frac{\boldsymbol{q}}{\| \boldsymbol{q} \|} + \beta \frac{\boldsymbol{d}_s}{\| \boldsymbol{d}_s \|}, \tag{16.11}$$

where β is a parameter for controlling the weight of the newly incorporated terms. Finally, the documents are ranked again according to the similarity $sim(\boldsymbol{d}_j, \boldsymbol{q}')$ to the expanded query.

This variation has two parameters: (1) the weighting parameter β, which defines how great the influence of the relevant documents is vs. the initial query and (2) the similarity threshold θ, which defines how many documents are used as positive feedback.

As opposed to the previously described approach, which defines a fixed amount of positive documents (n_1), the threshold θ describes only "how relevant" the documents must be in order to be used as positive feedback. Thus the number of documents used is dynamic and individual, depending on the document collection and on the current query. If many documents are similar to the initial query, then the document set $\boldsymbol{E}$ used for the expansion of the query is very large. But assuming the same θ, if only one document is sufficiently similar to the given query, then $\boldsymbol{E}$ contains only this single document.

16.2.4 Passage Retrieval

This section describes another important variant of the vector-space model called *passage retrieval*. It is closely related to the applications described in Sections 16.3.2 and 16.3.3.

Passage retrieval is a task where the goal is not to retrieve not whole documents but rather small snippets of text or *passages* relevant to the user's query. This technique has advantages over the standard vector-space model when documents consist of multiple topics. The standard vector-space model can be easily disturbed by the multi-topicality because the document vector is defined for the whole document. On the other hand, passage retrieval is less affected by this problem, since it is capable of retrieving relevant passages in documents. Another advantage of passage retrieval is that it is capable of segmenting relevant parts that interest

[1] Remark that the sum $\boldsymbol{d}_s$ is a single vector.

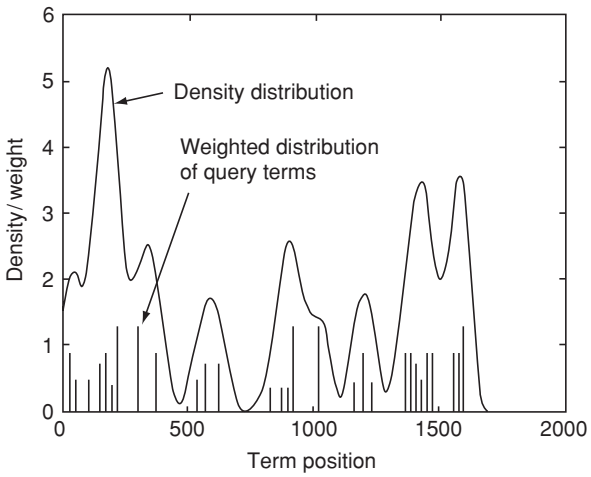

Fig. 16.2. Density distribution.

the user; it is unnecessary for the user to find where to read in retrieved documents.

In general, passages used in passage retrieval can be classified into three types: discourse, semantic and window passages [30]. Discourse passages are defined based on discourse units such as sentences and paragraphs. Semantic passages are obtained by segmenting text at the points where a topic of text changes. Window passages are determined based on the number of terms.

We have proposed a method called *density distributions* (DD) [29, 31, 32], which is based on window passages. A notion of density distributions was first introduced to locate explanations of a word in text [33] and applied to passage retrieval by some of the authors. Comparison of various windows for document retrieval and other tasks can be found in [34, 35].

An important idea of DD is the assumption that parts of documents with a particularly dense distribution of query terms are relevant to said query. Figure 16.2 shows an example of a density distribution. The horizontal axis indicates the positions of terms in a document. A weighted distribution of query terms in the document is shown as spikes on the figure; their height indicates the weight of a term. The density distribution shown in the figure is obtained by smoothing the spikes with a window function. The details are as follows.

Let us first introduce symbols used for explanations. The task of passage retrieval is to retrieve passages from a fixed set of documents or a document collection. In order to deal with documents as well as queries, both are represented using a set of terms. Let t_i $(1 \leq i \leq m)$ and d_j $(1 \leq j \leq n)$ be a term and a document in the database, respectively, where m is the number of different terms and n the number of documents in the database.

In passage retrieval, each document is viewed as a sequence of terms. Let $a_j(l)$ $(1 \leq l \leq L_j)$ be a term at a position l in a document d_j, where L_j is the document length measured in terms.

In general, some terms in a query are more important than others for locating appropriate passages. For example, the term *retrieval* in papers about information retrieval is ubiquitous and thus gives little evidence for distinguishing passages relevant to the query. In order to represent such information, we utilize the weight called *inverse document frequency* idf_i, which for any term t_i defined as

$$idf_i = \log \frac{n}{n_i}, \tag{16.12}$$

where n_i is the number of documents that include the term t_i.

Using the weight idf_i, the weighted distribution $b_j(l)$ of query terms is defined as

$$b_j(l) = w_{iq} \cdot idf_i, \tag{16.13}$$

where $a_j(l) = t_i$ and w_{iq} is the weight of a term t_i in a query. The weight w_{iq} is defined based on the frequency f_{iq} of the term t_i in the query as

$$w_{iq} = \log(f_{iq} + 1) \tag{16.14}$$

Smoothing of $b_j(l)$ enables us to obtain the density distribution $dd_j(l)$ for a document d_j:

$$dd_j(l) = \sum_{x=-W/2}^{W/2} f(x)\ b_j(l - x), \tag{16.15}$$

where $f(x)$ is a window function with a window size W. We employ the Hanning window function, whose shape is shown in Figure 16.3, and is defined by

$$f(x) = \begin{cases} \frac{1}{2}(1 + \cos 2\pi \frac{x}{W}) & \text{if } |x| \leq W/2; \\ 0 & \text{otherwise.} \end{cases} \tag{16.16}$$

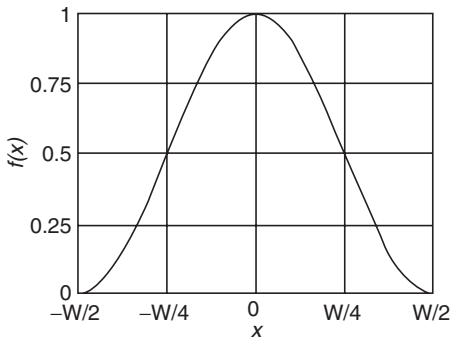

Fig. 16.3. Hanning window function.

16.3 Applications

In following section we propose three document retrieval methods based on the techniques described above: a collaborative information retrieval approach, a question-answering system and the recently developed document image question-answering system called *IQAS*.

16.3.1 Collaborative Information Retrieval

In this section, some central aspects of collaborative information retrieval in general are discussed and our approach for learning term-based concepts and query expansion techniques is introduced.

No Memory in Ad Hoc Search

A shortcoming of the ad hoc IR systems currently being used is their lack of any memory and their inability to learn. All information from a previous user or a previous query of the same user are gone immediately after the presentation of the list of relevant documents. Even systems with relevance feedback (see Section 16.2.3) do not learn. They only integrate the most recently given feedback information with the current query. All information about previous feedback is not stored and is unavailable for future queries, unless the feedback is explicitly coded within the new query.

Basic Idea of Collaborative Information Retrieval

To overcome this imperfection, we are using in our work an approach called *collaborative information retrieval* (CIR) [36–40]. This approach learns with the help of feedback information of *all* previous users (and also previous queries of the current user) to improve the retrieval performance with a lasting effect.

Generally, for satisfying the user's information needs, it is necessary to traverse a complex search process with many decisions and deliberations to achieve an optimal query. On the way from the initial query to a satisfying set of documents the user must invest a lot of effort and time that is lost in a common ad hoc IR system after the sending of the query. If another user (or the same user in the future) is searching for the same information, then they must invest the same time and effort again. In the case of complex search processes, it is virtually impossible to reconstruct all decisions and query reformulations in the same way. The users will not find the needed information (again) and will give up the search.

The idea of CIR is to store all information obtained during a search process in a repository and to use this information for future queries. A later user with similar information needs can profit from this automatically acquired wisdom on several ways:

- The current search process will be shortened and focussed on the desired topic.
- The retrieval performance of the IR system will be improved with the help of acquired wisdom of other users.

Restricted Scenario

This work is based on standard document collections (e.g. TREC [41]) to ensure comparability with standard retrieval methods described in literature. However, a shortcoming of these collections is that they do not contain any search processes as described above. They only contain a set of documents, a set of queries and the appropriate and relevant documents to each query (see also (16.17)). Due to this shortcoming, a restricted scenario is used for this work. The scenario has the following characteristics:

No complex search processes: The standard document collections only contain a set of documents, a set of user queries, and, for each query, a list of relevant documents.

No personalization: The scenario assumes one single, global user querying the IR system. No differentiation of several users, group profiles or personalization hierarchies are taken into consideration. (See [42] for further information about this topic.)

No user judgement: The user queries are qualitatively not differentiated and there is no judgement or assessment of the user or user queries.

No reflection over time: The queries are absolutely independent and no reflections or changes over time are made. An example problem that is not considered is the following: a user is initially searching for some publications to get an overview of a current problem. In the meantime, he is learning more and more about the topic and the next time he looks for more "deeper" documents and specific information, even though he is formulating the same query.

Global relevance feedback: The approaches shown here are based on global relevance feedback and learn with the complete learning set of queries, in contrast to ordinary local feedback that is used in every step within the search process to generate the new query. In our approach, the global feedback is used to directly form the new query in one single step.

Figure 16.4 illustrates the scenario used in this work. The user query q is transformed to the improved query q' with the help of relevance feedback information that is provided by the document collection.

A set of relevant documents is assigned to each query. Relevance information r_k for a query q_k is represented by an N-dimensional vector:

$$r_k = (r_{1k}, r_{2k}, \ldots, r_{Nk})^T, \quad 1 \leq k \leq L, \tag{16.17}$$

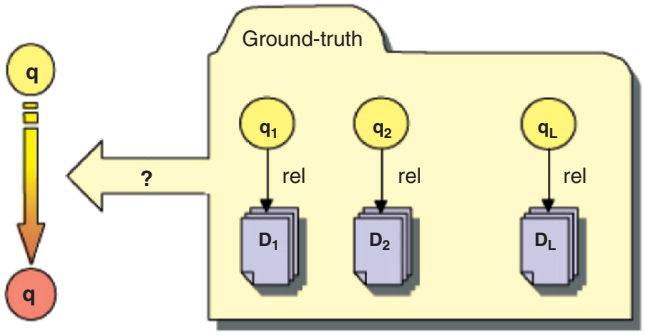

Fig. 16.4. Restricted scenario of the CIR with TREC collections.

where

$$r_{jk} = \begin{cases} 1 \text{ if document } j \text{ is relevant to query } k; \\ 0 \text{ otherwise.} \end{cases} \tag{16.18}$$

Here N is the number of documents and L the number of test queries in the collection. $()^T$ is the transpose of a matrix or a vector.

Central Aspects of Query-Expansion Techniques

The crucial point in query expansion is the question, "Which terms (or phrases) should be included in the query formulation?" If the query formulation is to be expanded by additional terms, there are two problems that are to be solved:

- How are these terms selected?
- How are the parameters estimated for these terms?

For the selection task, three different strategies have been proposed:

Dependent terms: Here, terms that are dependent on the query terms are selected. For this purpose, the similarity between all terms of the document collection has to be computed first [43].

Feedback terms: From the documents that have been judged by the user, the most significant terms (according to a measure that considers the distribution of a term within relevant and non-relevant documents) are added to the query formulation [19]. Clear improvements are reported in [19] and more recently in [44].

Interactive selection: By means of one of the methods mentioned before, a list of candidate terms is computed and presented to the user. The user then makes the final decision as to which terms are to be included in the query [3].

Many terms used in human communication are ambiguous or have several meanings [43], but most ambiguities are resolved automatically without noticing the ambiguity. The way this is done by humans is still an open problem in psychological research, but it is almost certain that the context in which a term occurs plays a central role [45, 46].

Historically, most attempts at automatically expanding queries have failed to improve the retrieval effectiveness, and it was often concluded that automatic query expansion based on statistical data was unable to bring a substantial improvement in the retrieval effectiveness [47]. This could be due to several reasons. Term-based query expansion approaches mostly rely upon hand-made thesauri or just plain co-occurrence data [48, 49]. They do not use learning technologies for the query terms. On the other hand, those that use learning technologies (neural networks, support vector machines, etc.) are query based. This means these systems learn concepts (or additional terms) for the complete query.

The vital advantage of using term-based concepts and not learning the complete query is that other users can profit from the learned concepts. A statistical evaluation of log files has shown that the probability that a searcher uses exactly the same query as a previous searcher is much lower than the probability that parts of a query (phrases or terms) occur in other queries. So, even if a web searcher never uses a given search term, the probability that some other searcher has used it is very high, and the former can therefore profit from the learned concept.

Learning Term-based Concepts

A problem of the standard VSM is that a query is often too short to rank documents appropriately. To cope with this problem, one approach is to enrich the original query with terms that occur in the documents of the collection.

Our method uses feedback information and information globally available from previous queries. Feedback information in our environment is available within the ground truth data provided by the test document collections. The ground truth provides relevance information – i.e. for each query, a list of relevant documents exists.

In contrast to traditional pseudo-relevance feedback methods where the top j-ranked documents are assumed to be relevant and their terms incorporated into the expanded query, we use a different technique to compute relevant documents [38]. The approach is divided into two phases, as shown in Figure 16.5:

- The *learning phase* for each term works as follows
  1. Select old queries in which the specific query term occurs.
  2. From these selected old queries, get the sets of relevant documents from the ground truth data.

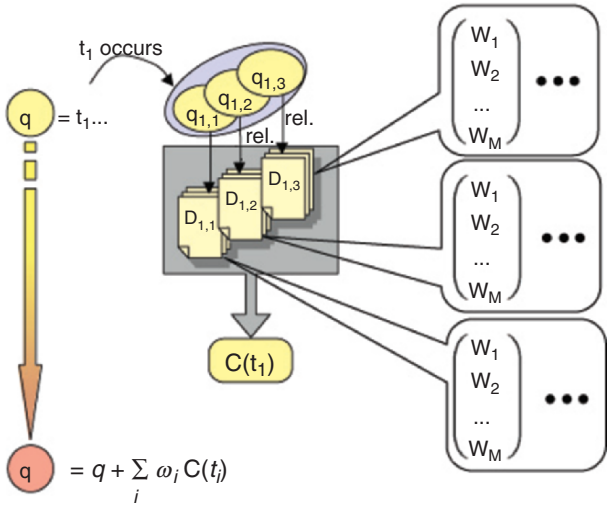

Fig. 16.5. Strategy of learning term-based concepts.

3. From each set of relevant documents, compute a new document vector and use these document vectors to build the term concept.

- The *expansion phase* for each term is then performed as documented in literature
  1. Select the appropriate concept of the current term.
  2. Use a weighting scheme to enrich the new query with the concept.

For the formal description of the learning phase, we need the following definitions:

- $\mathbf{D} = \boldsymbol{d}_1, \ldots, \boldsymbol{d}_N$ is the set of all documents.
- $\boldsymbol{Q} = \boldsymbol{q}_1, \ldots, \boldsymbol{q}_L$ is the set of all known queries.
- $\boldsymbol{q}_k = (w_{1k}, \ldots, w_{ik}, \ldots, w_{Mk})^T$ is represented within the vector-space model. For each term of the query, the appropriate weight w_{ik} is between 0 and 1.
- $\boldsymbol{R}^+(\boldsymbol{q}_k) = \{\boldsymbol{d}_j \in D \mid r_{ij} = 1\}$ is the set of all documents relevant to the query $\boldsymbol{q}_k$ (see also (16.18)).

Now, the first step of the learning phase collects all queries having the ith term in common:[2]

$$\boldsymbol{Q}_i = \{\boldsymbol{q}_k \in \boldsymbol{Q} \mid w_{ik} \neq 0\}\,. \tag{16.19}$$

Step two collects all documents that are relevant to these collected queries:

$$\mathbf{D}_{ik} = \left\{\ \boldsymbol{d}_j \ \mid\ \boldsymbol{d}_j \in \boldsymbol{R}^+(\boldsymbol{q}_k) \wedge \boldsymbol{q}_k \in \boldsymbol{Q}\ \right\}. \tag{16.20}$$

[2] If the ith term does not occur in any query $\boldsymbol{q}_k$, then the set $\boldsymbol{Q}_i$ is empty.

In the last step of the learning phase, the concept of each ith term is built as the sum of all documents (i.e. vectors of term weights) that are relevant to the known queries that have the term in common:

$$\mathbf{C}_i = \sum_{\boldsymbol{d}_j \in \mathbf{D}_{ik}} \boldsymbol{d}_j. \tag{16.21}$$

As with queries and documents, a concept is represented by a vector of term weights. If no query $\boldsymbol{q}_k$ contains term i, then the corresponding concept $\mathbf{C}_i$ is represented as $(0, \ldots, 0)^T$.

Now that the term-based concepts have been learned, the user query $\boldsymbol{q}$ can be expanded term by term. Thus, the expanded query vector $\boldsymbol{q}'$ is obtained by

$$\boldsymbol{q}' = \boldsymbol{q} + \sum_{i=1}^{M} \omega_i \mathbf{C}_i, \tag{16.22}$$

where ω_i are parameters for weighting the concepts. In the experiments described below, ω_i is globally set to 1.

Before applying the expanded query, it is normalized by

$$\boldsymbol{q}'' = \frac{\boldsymbol{q}'}{||\boldsymbol{q}'||}. \tag{16.23}$$

For this approach, the complete documents (all term weights w_{ij} of the relevant documents) are summed up and added to the query. Although in literature it is reported that using just the top-ranked terms is sufficient or sometimes better, experiments with this approach on the TREC collections have shown that the more words used to learn the concepts, the better the results. So the decision was made to always use the complete documents and not merely some (top-ranked) terms.

If no ground truth of relevant documents is available, (pseudo-)relevance feedback techniques can be used and the concepts are learned by adding terms from the retrieved relevant documents.

The advantage of the document transformation approach – leaving permanent effects on a system – also holds for learned concepts.

16.3.2 Question-Answering System

Next we show an important application of passage retrieval called *question-answering* (QA). The task of QA is to find a text portion that contains an answer to a given query. A typical interaction between a user and a QA system might go as follows:

User: What is the population of Japan?

System: ...Ministry said last month Japanese population was 122.74 million at...

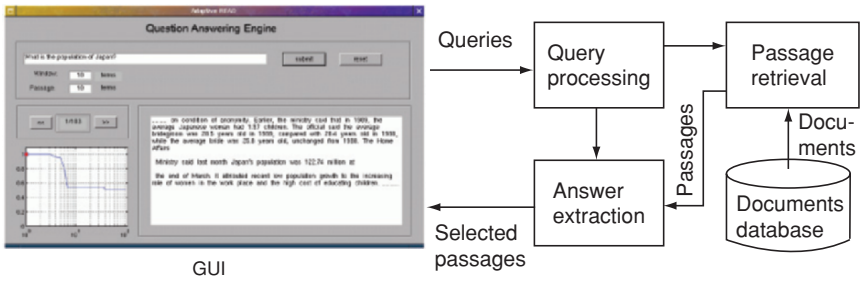

Fig. 16.6. Question answering system.

In order to realize this scenario, it is necessary to apply a method of finding passages that include answers to the question. This is why the QA task is an important application area of passage retrieval.

Overview of the System

Figure 16.6 illustrates an overview of the proposed QA system [50]. The system employs three steps of processing: query processing, passage retrieval and answer extraction. The outline of processing is as follows. First, the system takes as input a natural-language query from a user through the GUI. Next, the query is sent to query processing for extracting query terms and finding the types of the query. The extracted query terms are sent to passage retrieval for retrieving passages from documents in the database. Retrieved passages are then transferred to the answer extraction component for examining whether or not they include entities required by the query. Lastly, filtered passages are sorted according to their scores, and the top-ranking passage is displayed on the GUI.

Types of Queries and Terms

An important point specific to the QA task is in the query processing and the answer extraction: some "types" are assigned to terms as well as to queries for the purpose of selecting appropriate passages. These types are employed for clarifying the following points:

- What does a query ask about? (types of queries)
- Which terms can be a part of an answer to the query? (types of terms)

We currently use the types shown in Table 16.1 for both queries and terms. Note that more than one type can be assigned to a term as well as to a query, while some terms and queries have no type whatsoever. In the case that a query has a type, passages that contain the terms of the same type can be possible answers.

Processing Steps

As shown in Figure 16.6, the processing of the system is threefold: (1) query processing, (2) passage retrieval and (3) answer extraction.

Query Processing

The first step, *query processing*, is to analyse the query to obtain its query terms and types. In order to identify types of the query, a pattern-matching technique for finding typical patterns (such as those shown in Table 16.1) is applied. In parallel, stemming and stopword elimination are applied to the query to obtain query terms. Extracted query terms are utilized to find relevant passages at the next step.

Passage Retrieval

We employ the passage retrieval method of density distributions described in Section 16.2.4. The outline of processing is as follows. First, density distributions such as in Figure 16.2 are calculated for all documents. Passages are then extracted based on every peak in the density distributions. For example, in the density distribution shown in Figure 16.7, there are two peaks, P_1 and P_2. For each peak, a passage with fixed width (P_w) is extracted by taking the position of a peak as a centre of the passage. Each passage has its score defined as the density of its peak. In Figure 16.7, S_1 and S_2 are the scores. Finally, passages extracted from all density distributions are ranked according to their scores. Note that we do not take

Table 16.1. Types of queries and terms

Type	Examples of queries	Examples of terms
Money	`How much money...?`	`$10`
Time	`How long...?`	`10:00`, `a.m.`, `p.m.`
Month	`What time of year...?`	`Jan.`, `December`
Number	`How far...?`	`10,000`, `0.5`
Name	`Who...?`	`Bush`

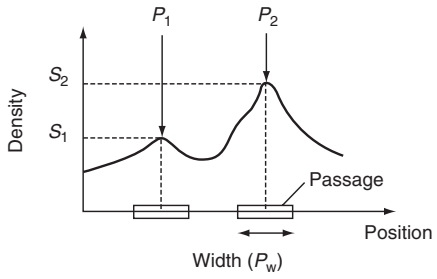

Fig. 16.7. Passages extracted from a density distribution.

account of types of queries and terms in extracting passages. At this step, passages are obtained based solely on the density of terms in the query.

Answer Extraction

This step is to filter out inappropriate passages using the types. The filtering strategy is simple: if types are assigned to a query, passages that include at least one term of one of the types are retained, but passages without such terms are removed from the ranked list of passages.

16.3.3 Document Image Retrieval

Question-Answering for Document Images

All of the methods described so far are for electronic documents. However, it is not enough for us to deal only with electronic documents, at least because of the following two reasons. First, we now have a huge number of document images in various databases and digital libraries. For example, the journal *Communications of the ACM* in the ACM digital library [51] consists of 80% document images and 20% electronic documents. Another reason is that mobile devices with digital cameras are now coming into common use. The resolution of such cameras has been rapidly improved in recent years.[3] Some users have started using them not only for imaging faces and scenes, but also for taking digital copies of documents, because it is much more convenient than writing a memo or finding a photocopier.[4] This indicates that not only legacy documents but also new documents continue to be stored as document images.

In order to utilize such document images, establishing a way of retrieving them is required. As described in the previous section, it would be good to implement a question-answer system for document images for improving the usability of retrieval. From this viewpoint, we have developed a QA system for document images called *IQAS*[5] (Document **I**mage ***Q***uestion **A**nswering **S**ystem) [52].

Figure 16.8 illustrates the graphical user interface of IQAS. In this figure, the part relevant to the query is highlighted. The user can magnify the retrieved part in the page. If it does not contain the answer, the user can obtain the next page.

[3] Early in 2005, a camera phone with the resolution of seven megapixels was released. The size of captured images is equivalent to the size of images obtained by scanning a sheet of A4 paper at 270 dpi.

[4] In Japan, "digital shoplifting" – i.e. making pictures of books and magazines in bookstores with mobile camera phones – has become a subject of economic concern.

[5] The pronunciation of the term IQAS is close to the Japanese word *ikasu*, which can mean either *to exploit* or *cool*.

Fig. 16.8. Graphical user interface.

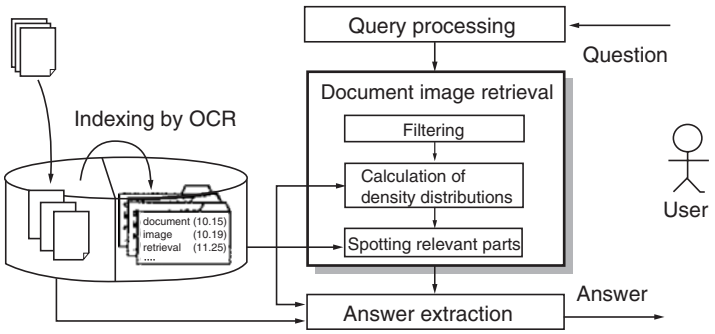

Fig. 16.9. System configuration.

Figure 16.9 shows the system configuration of IQAS that follows the configuration shown in the previous section except for the "document image retrieval" part. In the step of document image retrieval, we utilize a passage retrieval technique based on the density distributions extended for dealing with two-dimensional images.

Document Image Retrieval in IQAS

As shown in Figure 16.9, the step of document image retrieval consists of three smaller steps: (1) filtering, (2) calculating density distributions and (3) identifying relevant parts. In the following, the details of each step are explained after a brief introduction of indexing of document images and query processing.

Indexing

Since document images are two-dimensional entities, it is necessary to extend the indexing scheme to be capable of dealing with two-dimensionality.

In our indexing scheme, document images are viewed as two-dimensional distributions of terms.

The outline of indexing is as follows. First, all words and their bounding boxes are extracted from page images with the help of OCR. Second, stemming and stopword elimination are applied to the extracted words. The resultant words are called *terms* and stored with the centres of their bounding boxes.

Query Processing

The task of query processing is shared by the QA system for electronic documents. Suppose we have a query, "Where is the Baseball Hall of Fame?". The query type is "location" and the query terms are "baseball", "hall", and "fame". Note that only the extraction of query terms is relevant to the task of document image retrieval. In the following, the sequence of extracted query terms is called *the query* and represented as $q = (q_1, \ldots, q_r)$, where q_i is called a query term and i indicates the order of occurrence in the query. For the above example, $q = (\text{baseball}, \text{hall}, \text{fame})$.

Filtering

Filtering is first applied to ease the burden of the next step, which is relatively time consuming. The task here is to select N_v pages that are likely to include the answer to the query. For this purpose, we utilize the simple vector-space model (VSM). Pages are sorted according to similarity, and the top N_v pages are selected and delivered to the next step.

Calculation of Density Distributions

This step calculates density distributions of the query q for each selected page. Density distributions of the query are defined based on those of each query term q_i. Let $T_i^{(p)}(x, y)$ be a weighted distribution of a term $q_i (= t_k)$ in a selected page p defined by

$$T_i^{(p)}(x, y) = \begin{cases} idf_k & \text{if } q_i (= t_k) \text{ occurs at } (x, y), \\ 0 & \text{otherwise}, \end{cases} \tag{16.24}$$

where (x, y) is the centre of the bounding box of a term. A density distribution $D_i^{(p)}(x, y)$ is a weighted distribution of q_i smoothed by a window $W(x, y)$:

$$D_i^{(p)}(x, y) = \sum_{u=-M_x/2}^{M_x/2} \sum_{v=-M_y/2}^{M_y/2} W(x - u, y - v)\, T_i^{(p)}(u, v). \tag{16.25}$$

As a window function, we utilize a pyramidal function with the window widths M_x (the horizontal width) and M_y (the vertical width) shown in Figure 16.10.

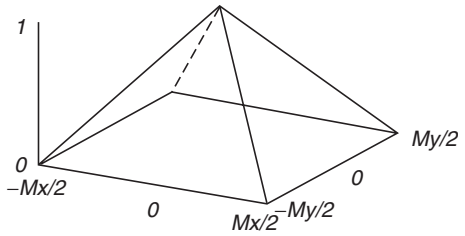

Fig. 16.10. Window function.

A way to define the density distributions of the whole query is to sum up $D_i^{(p)}$ for all q_i, as described in Section 16.2.4. However, this sometimes causes a problem of insufficient discrimination power in the case of document image retrieval: irrelevant parts are erroneously retrieved due to the effect of frequent terms. Therefore, in order to improve the discrimination power, we introduce a notion of N-grams in calculating the distributions. To be precise, we consider the smaller number of successive query terms. For example, the density distribution obtained by r–1 successive query terms is defined by

$$C_{r-1}^{(p)}(x,y) = \prod_{i=1}^{r-1} D_i^{(p)}(x,y) + \prod_{i=2}^{r} D_i^{(p)}(x,y), \tag{16.26}$$

where r is the number of query terms. For the general case of k successive query terms, the density distribution is defined by

$$C_k^{(p)}(x,y) = \sum_{j=0}^{r-k} \prod_{i=j+1}^{j+k} D_i^{(p)}(x,y). \tag{16.27}$$

In the proposed method, the density distribution of the whole query for a page p is defined as the weighted sum of the combinations from all the r terms down to s successive terms:

$$D^{(p)}(x,y) = \sum_{k=s}^{r} \alpha_k C_k^{(p)}(x,y), \tag{16.28}$$

where the parameter s and the weight α_k are experimentally determined.

Identifying Relevant Parts

Based on the density distribution of (16.28), parts that are likely to include the answer are located on page images. First, page images are ranked according to their score of the maximum density:

$$s^{(p)} = \max_{x,y} D^{(p)}(x,y). \tag{16.29}$$

Then, the top-ranked page is presented to the user through the GUI shown in Figure 16.8.

16.4 Summary and Conclusion

In this chapter, we have given an overview of the vector-space model and basic techniques commonly utilized in document information retrieval. Relevance feedback techniques rely on user feedback regarding the relevance of retrieved documents, whereas passage retrieval is based on the assumption that not the document as a whole but rather specific passages are relevant to a query if they contain terms of the query with high density.

Collaborative information retrieval systems are document retrieval systems that adapt to the user's needs by transferring search knowledge from one user to others. For this, our new query expansion method is learning term-based concepts from previous queries and their relevant documents. In contrast to other relevance feedback and query expansion methods, the query is expanded by adding those terms that are similar to the *concept* of individual query terms, rather than selecting terms that are similar to the complete query or that are similar to single query terms.

For question answering, the passage retrieval method is applied for finding very short snippets of documents that contain an answer to a given query.

In order to use the passage retrieval method as a tool of intelligent access to documents, the following points should be further considered. First, the window size appropriate for analysing documents should be determined automatically. Second, sophisticated learning techniques must be incorporated in order to avoid manual adjustment of processing parameters, including expressions for the type identification in the QA task.

Retrieving document images is still an active research topic and has become increasingly an object of public concern. For finding relevant images or parts thereof, question-answering and passage retrieval techniques are combined for improving the usability of retrieval.

The issues described in this chapter will be subjects of our future research.

References

1. Cleverdon, C.W. (1984). Optimizing convenient online access to bibliographic databases. *Information Services and Use, 4*, pp. 37–47.
2. Salton, G. (1971). *The SMART Retrieval System – Experiments in Automatic Document Processing*. Englewood Cliffs, NJ: Prentice-Hall, Inc.
3. Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Reading, MA: Addison-Wesley.
4. Ferber, R. (2003). *Information Retrieval – Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. Germany: dpunkt.verlag.
5. Luhn, H.P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development, 2,* pp. 159–165.

6. Porter, M.F. (1980). An algorithm for suffix stripping. *Program, 14*, pp. 130–137.
7. Porter, M.F. (2005). http://www.tartarus.org/_martin/PorterStemmer/.
8. Porter, M.F. (2005). A small string processing language for creating stemmers. http://snowball.tartarus.org.
9. Porter, M.F. (1983). Information retrieval at the Sedgwick Museum. *Information Technology: Research and Development, 2*, pp. 169–186.
10. Lewis, D.D. and Spärck Jones, K. (1993). Natural language processing for information retrieval. *Technical Report 307, University of Cambridge Computer Laboratory.*
11. Kupiec, J., Kimber, D., and Balasubramanian, V. (1994). Speech-based retrieval using semantic co-occurrence filtering. *Proceedings of the Human Language Technology (HLT) Conference, US Advanced Projects Research Agency (ARPA)*, pp. 373–377.
12. Brauen, T.L. (1971). *Document Vector Modification.* Englewood Cliffs, NJ: Prentice Hall, pp. 456–484.
13. Salton, G. and Lesk, M. (1968). Computer evaluation of indexing and text processing. *Journal of the ACM, 15*, pp. 8–36.
14. Belkin, N.J. and Croft, W.B. (1987). Retrieval techniques. *Annual Review of Information Science and Technology, 22*, pp. 109–145.
15. Harman, D.K. (1992). *Ranking Algorithms.* Upper Saddle River, NJ: Prentice Hall, pp. 363–392.
16. Salton, G., Allen, J., and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management, 24*, pp. 513–523.
17. Fuhr, N. and Buckley, C. (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems, 9*, pp. 223–248.
18. Turtle, H.R. and Croft, W.B. (1991). Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems, 9*, pp. 187–222.
19. Rocchio, J.J. *Relevance Feedback in Information Retrieval.* Englewood Cliffs, NJ: Prentice Hall, pp. 313–323.
20. Wilson, R. and Martinez, T.R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research, 6*, pp. 1–34.
21. Kemp, C. and Ramamohanarao, K. (2002). Long-term learning for web search enginges. In: T. Elomaa, H. Mannila, and H. Toivonen (Eds.). *Proceedings of the Sixth European Conference of Principles of Data Mining and Knowledge Discovery (PKKD2002).* Lecture Notes in Artificial Intelligence 2431, Helsinki, Finland, Springer, pp. 263–274.
22. Bhuyan, J.N., Deogun, J.S., and Raghavan, V.V. (1997). An adaptive information retrival system based on user-oriented clustering. *ACM Transaction on Information Systems.*
23. Gudivada, V.N., Raghavan, V.V., Grosky, W.I., and Kasanagottu, R. (1997). Information retrieval on the World Wide Web. *IEEE Internet Computing, 1.*
24. Friedman, S.R., Maceyak, J.A., and Weiss, S.F. (1971). *A Relevance Feedback System Based on Document Transformations.* Englewood Cliffs, NJ: Prentice Hall, pp. 447–455.
25. Salton, G. (1989). *Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer.* Reading, MA: Addison-Wesley.

26. Savoy, J. and Vrajitoru, D. (1996). Evaluation of learning schemes used in information retrieval. *Technical Report CR-I-95-02, Faculty of Sciences, University of Neuchâtel*.
27. Rocchio, J.J. (1966). Document retrieval systems – optimization and evaluation. Ph.D. thesis. Cambridge, MA: Harvard Computational Laboratory.
28. Salton, G. and Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the ASIS, 41*, pp. 288–297.
29. Kise, K., Junker, M., Dengel, A., and Matsumoto, K. (2001). Passage-based document retrieval as a tool for text mining with user's information needs. In: K.P. Jantke, A. Shinohara (Eds.). *Discovery Science*. Lecture Notes in Computer Science. Princeton, NJ: Springer, Volume 2226, pp. 155–169.
30. Callan, J.P. (1994). Passage-level evidence in document retrieval. In: W.B. Croft, C.J. Rijsbergen (Eds.). *SIGIR*. New York: ACM/Springer, pp. 302–310.
31. Kise, K., Mizuno, H., Yamaguchi, M., and Matsumoto, K. (1999). On the use of density distribution of keywords for automated generation of hypertext links from arbitrary parts of documents. *ICDAR*, pp. 301–304.
32. Kise, K., Junker, M., Dengel, A., and Matsumoto, K. (2001). Experimental evaluation of passage-based document retrieval. *ICDAR*. Silver Spring, MD: IEEE Computer Society, pp. 592–596.
33. Kurohashi, S., Shiraki, N., and Nagao, M. (1997). A method for detecting important descriptions of a word based on its density distribution in text. *Transactions of Information Processing Society of Japan, 38*, pp. 845–853 (In Japanese).
34. Kretser, O. and Moffat, A. (1999). Effective document presentation with a locality-based similarity heuristic. *SIGIR*. New York: ACM, pp. 113–120.
35. Kozima, H. and Furugori, T. (1994). Segmenting narrative text into coherent scenes. *Literary and Linguistic Computing, 9*, pp. 13–19.
36. Hust, A., Klink, S., Junker, M., and Dengel, A. (2003). Towards collaborative information retrieval: three approaches. *Text Mining*, pp. 97–112.
37. Klink, S. (2004). Improving document transformation techniques with collaborative learned term-based concepts. *Reading and Learning*, pp. 281–305.
38. Klink, S., Hust, A., and Junker, M. (2002). TCL – an approach for learning meanings of queries in information retrieval systems. *Content Management – Digitale Inhalte als Bausteine einer vernetzten Welt*, pp. 15–25.
39. Klink, S., Hust, A., Junker, M., and Dengel, A. (2002). Collaborative learning of term-based concepts for automatic query expansion. *Proceedings of the 13th European Conference on Machine Learning (ECML 2002)*. Lecture Notes in Artificial Intelligence. Helsinki, Finland: Springer, Volume 2430, pp. 195–206.
40. Klink, S., Hust, A., Junker, M., and Dengel, A. (2002). Improving document retrieval by automatic query expansion using collaborative learning of term-based concepts. *Proceedings of the Fifth International Workshop on Document Analysis Systems (DAS 2002)*. Lecture Notes in Computer Science. Princeton, NJ: Springer, Volume 2423, pp. 376–387.
41. Text REtrieval conference (TREC). (2005). http://trec.nist.gov/.
42. Klink, S. (2001). Query reformulation with collaborative concept-based expansion. *Proceedings of the First International Workshop on Web Document Analysis (WDA 2001), Seattle, Washington, USA*, pp. 19–22.

43. Pirkola, A. (1999). Studies on Linguistic Problems and Methods in Text Retrieval: The Effects of Anaphor and Ellipsis Resolution in Proximity Searching, and Translation and Query Structuring Methods in Cross-Language Retrieval. Doctoral dissertation. Finland: Department of Information Science, University of Tampere.
44. Jansen, B.J., Spink, A., and Saracevic, T. (2000). Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management, 36*, pp. 207–227.
45. Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics, 24*, pp. 97–123.
46. Oh, J.H. and Choi, K.S. (2002). Word sense disambiguation using static and dynamic sense vectors. *Proceedings of the 19th International Conference on Computational Linguistics, Taipei, Taiwan*.
47. Peat, H.J. and Willet, P. (1991). The limitations of term cooccurrence data for query expansion in document retrieval systems. *Journal of the American Society of Information Systems, 42*, pp. 378–383.
48. Chen, J.N. and Chang, J.S. (1998). A concept-based adaptive approach to word sense disambiguation. *Proceedings of 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*. Los Altos, CA: Morgan Kaufmann, Volume 1, pp. 237–243.
49. Guthriee, J.A., Guthrie, L., Aidinejad, H., and Wilks, Y. (1991). Subject-dependent cooccurrence and word sense disambiguation. *Proceedings of 29th Annual Meeting of the Association for Computational Linguistics*, pp. 146–152.
50. Kise, K., Junker, M., Dengel, A., and Matusmoto, K. (2004). Passage retrieval based on density distributions of terms and its applications to document retrieval and question answering. *Reading and Learning, Adaptive Content Recognition*. Lecture Notes in Computer Science. Springer, Volume 2956, pp. 306–327.
51. The ACM ditigal library. (2005). http://www.acm.org/dl/.
52. Kise, K., Fukushima, S., and Matsumoto, K. (2004). Document image retieval in a question answering system for document images. *Proceedings of the Sixth International Workshop on Document Analysis Systems (DAS 2004)*. Lecture Notes in Computer Science. Springer, Volume 3163, pp. 521–532.