Chapter 5
Query Operations

5.1 Introduction

Without detailed knowledge of the collection make-up and of the retrieval environment, most users find it difficult to formulate queries which are well designed for retrieval purposes. In fact, as observed with Web search engines, the users might need to spend large amounts of time reformulating their queries to accomplish effective retrieval. This difficulty suggests that the first query formulation should be treated as an initial (naive) attempt to retrieve relevant information. Following that, the documents initially retrieved could be examined for relevance and new improved query formulations could then be constructed in the hope of retrieving additional useful documents. Such query reformulation involves two basic steps: expanding the original query with new terms and reweighting the terms in the expanded query.

In this chapter, we examine a variety of approaches for improving the initial query formulation through query expansion and term reweighting. These approaches are grouped in three categories: (a) approaches based on feedback information from the user; (b) approaches based on information derived from the set of documents initially retrieved (called the *local* set of documents); and (c) approaches based on global information derived from the document collection. In the first category, user relevance feedback methods for the vector and probabilistic models are discussed. In the second category, two approaches for local analysis (i.e., analysis based on the set of documents initially retrieved) are presented. In the third category, two approaches for global analysis are covered.

Our discussion is not aimed at completely covering the area, neither does it intend to present an exhaustive survey of query operations. Instead, our discussion is based on a selected bibliography which, we believe, is broad enough to allow an overview of the main issues and tradeoffs involved in query operations. Local and global analysis are highly dependent on clustering algorithms. Thus, clustering is covered throughout our discussion. However, there is no intention of providing a complete survey of clustering algorithms for information retrieval.

5.2 User Relevance Feedback

Relevance feedback is the most popular query reformulation strategy. In a relevance feedback cycle, the user is presented with a list of the retrieved documents and, after examining them, marks those which are relevant. In practice, only the top 10 (or 20) ranked documents need to be examined. The main idea consists of selecting important terms, or expressions, attached to the documents that have been identified as relevant by the user, and of enhancing the importance of these terms in a new query formulation. The expected effect is that the new query will be moved towards the relevant documents and away from the non-relevant ones.

Early experiments using the Smart system [695] and later experiments using the probabilistic weighting model [677] have shown good improvements in precision for small test collections when relevance feedback is used. Such improvements come from the use of two basic techniques: query expansion (addition of new terms from relevant documents) and term reweighting (modification of term weights based on the user relevance judgement).

Relevance feedback presents the following main advantages over other query reformulation strategies: (a) it shields the user from the details of the query reformulation process because all the user has to provide is a relevance judgement on documents; (b) it breaks down the whole searching task into a sequence of small steps which are easier to grasp; and (c) it provides a controlled process designed to emphasize some terms (relevant ones) and de-emphasize others (non-relevant ones).

In the following three subsections, we discuss the usage of user relevance feedback to (a) expand queries with the vector model, (b) reweight query terms with the probabilistic model, and (c) reweight query terms with a variant of the probabilistic model.

5.2.1 Query Expansion and Term Reweighting for the Vector Model

The application of *relevance feedback* to the vector model considers that the term-weight vectors of the documents identified as relevant (to a given query) have similarities among themselves (i.e., relevant documents resemble each other). Further, it is assumed that non-relevant documents have term-weight vectors which are dissimilar from the ones for the relevant documents. The basic idea is to reformulate the query such that it gets closer to the term-weight vector space of the relevant documents.

Let us define some additional terminology regarding the processing of a given query q as follows,

D_r: set of relevant documents, as identified by the user, among the *retrieved* documents;
D_n: set of non-relevant documents among the *retrieved* documents;
C_r: set of relevant documents among all documents in the collection;

$|D_r|, |D_n|, |C_r|$: number of documents in the sets D_r, D_n, and C_r, respectively;
α, β, γ: tuning constants.

Consider first the unrealistic situation in which the complete set C_r of relevant documents to a given query q is known in advance. In such a situation, it can be demonstrated that the best query vector for distinguishing the relevant documents from the non-relevant documents is given by,

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\forall \vec{d}_j \in C_r} \vec{d}_j - \frac{1}{N - |C_r|} \sum_{\forall \vec{d}_j \notin C_r} \vec{d}_j \qquad \textbf{(5.1)}$$

The problem with this formulation is that the *relevant* documents which compose the set C_r are not known a priori. In fact, we are looking for them. The natural way to avoid this problem is to formulate an initial query and to incrementally change the initial query vector. This incremental change is accomplished by restricting the computation to the documents *known* to be relevant (according to the user judgement) at that point. There are three classic and similar ways to calculate the modified query $\vec{q}_m$ as follows,

$$\begin{aligned}
Standard\_Rochio: \quad \vec{q}_m &= \alpha\,\vec{q} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j \\
Ide\_Regular: \quad \vec{q}_m &= \alpha\,\vec{q} + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j \\
Ide\_Dec\_Hi: \quad \vec{q}_m &= \alpha\,\vec{q} + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma\; max_{non\text{-}relevant}(\vec{d}_j)
\end{aligned}$$

Expan[illegible]
is
occurr[illegible]

where $max_{non-relevant}(\vec{d}_j)$ is a reference to the highest ranked non-relevant document. Notice that now D_r and D_n stand for the sets of relevant and non-relevant documents (among the retrieved ones) according to the user judgement, respectively. In the original formulations, Rochio [678] fixed $\alpha = 1$ and Ide [391] fixed $\alpha = \beta = \gamma = 1$. The expressions above are modern variants. The current understanding is that the three techniques yield similar results (in the past, Ide Dec-Hi was considered slightly better).

The Rochio formulation is basically a direct adaptation of equation 5.1 in which the terms of the original query are added in. The motivation is that in practice the original query q may contain important information. Usually, the information contained in the relevant documents is more important than the information provided by the non-relevant documents [698]. This suggests making the constant γ smaller than the constant β. An alternative approach is to set γ to 0 which yields a *positive* feedback strategy.

The main advantages of the above relevance feedback techniques are simplicity and good results. The simplicity is due to the fact that the modified term weights are computed directly from the set of retrieved documents. The good

5.2.4 Evaluation of Relevance Feedback Strategies

Consider the modified query vector $\vec{q}_m$ generated by the Rochio formula and assume that we want to evaluate its retrieval performance. A simplistic approach is to retrieve a set of documents using $\vec{q}_m$, to rank them using the vector formula, and to measure recall-precision figures relative to the set of relevant documents (provided by the experts) for the original query vector $\vec{q}$. In general, the results show spectacular improvements. Unfortunately, a significant part of this improvement results from the higher ranks assigned to the set R of documents

already identified as relevant during the feedback process [275]. Since the user has seen these documents already (and pointed them as relevants), such evaluation is unrealistic. Further, it masks any real gains in retrieval performance due to documents not seen by the user yet.

A more realistic approach is to evaluate the retrieval performance of the modified query vector $\vec{q}_m$ considering only the *residual collection* i.e., the set of all documents minus the set of feedback documents provided by the user. Because highly ranked documents are removed from the collection, the recall-precision figures for $\vec{q}_m$ tend to be lower than the figures for the original query vector $\vec{q}$. This is not a limitation because our main purpose is to compare the performance of distinct relevance feedback strategies (and not to compare the performance before and after feedback). Thus, as a basic rule of thumb, any experimentation involving relevance feedback strategies should always evaluate recall-precision figures relative to the residual collection.

5.3 Automatic Local Analysis

In a user relevance feedback cycle, the user examines the top ranked documents and separates them into two classes: the relevant ones and the non-relevant ones. This information is then used to select new terms for query expansion. The reasoning is that the expanded query will retrieve more relevant documents. Thus, there is an underlying notion of *clustering* supporting the feedback strategy. According to this notion, known relevant documents contain terms which can be used to describe a larger cluster of relevant documents. In this case, the description of this larger cluster of relevant documents is built interactively with assistance from the user.

A distinct approach is to attempt to obtain a description for a larger cluster of relevant documents automatically. This usually involves identifying terms which are related to the query terms. Such terms might be synonyms, stemming variations, or terms which are close to the query terms in the text (i.e., terms with a distance of at most k words from a query term). Two basic types of strategies can be attempted: global ones and local ones.

In a global strategy, all documents in the collection are used to determine a global thesaurus-like structure which defines term relationships. This structure is then shown to the user who selects terms for query expansion. Global strategies are discussed in section 5.4.

In a local strategy, the documents retrieved for a given query q are examined at query time to determine terms for query expansion. This is similar to a relevance feedback cycle but might be done without assistance from the user (i.e., the approach might be fully automatic). Two local strategies are discussed below: local clustering and local context analysis. The first is based on the work done by Attar and Fraenkel in 1977 and is used here to establish many of the fundamental ideas and concepts regarding the usage of clustering for query expansion. The second is a recent work done by Xu and Croft in 1996 and illustrates the advantages of combining techniques from both local and global analysis.

5.3.1 Query Expansion Through Local Clustering

Adoption of clustering techniques for query expansion is a basic approach which has been attempted since the early years of information retrieval. The standard approach is to build global structures such as association matrices which quantify term correlations (for instance, number of documents in which two given terms co-occur) and to use correlated terms for query expansion. The main problem with this strategy is that there is not consistent evidence that global structures can be used effectively to improve retrieval performance with general collections. One main reason seems to be that global structures do not adapt well to the local context defined by the current query. One approach to deal with this effect is to devise strategies which aim at optimizing the current search. Such strategies are based on *local clustering* and are now discussed. Our discussion is based on the original work by Attar and Fraenkel which appeared in 1977.

We first define basic terminology as follows.

Definition *Let $V(s)$ be a non-empty subset of words which are grammatical variants of each other. A canonical form s of $V(s)$ is called a* stem. *For instance, if $V(s)$={polish,polishing,polished} then s=polish.*

For a detailed discussion on stemming algorithms see Chapter 7. While stems are adopted in our discussion, the ideas below are also valid for non-stemmed keywords. We proceed with a characterization of the local nature of the strategies covered here.

Definition *For a given query q, the set D_l of documents retrieved is called the local document set. Further, the set V_l of all distinct words in the local document set is called the local vocabulary. The set of all distinct stems derived from the set V_l is referred to as S_l.*

We operate solely on the documents retrieved for the current query. Since it is frequently necessary to access the text of such documents, the application of local strategies to the Web is unlikely at this time. In fact, at a client machine, retrieving the text of 100 Web documents for local analysis would take too long, reducing drastically the interactive nature of Web interfaces and the satisfaction of the users. Further, at the search engine site, analyzing the text of 100 Web documents would represent an extra spending of CPU time which is not cost effective at this time (because search engines depend on processing a high number of queries per unit of time for economic survival). However, local strategies might be quite useful in the environment of intranets such as, for instance, the collection of documents issued by a large business company. Further, local strategies might also be of great assistance for searching information in specialized document collections (for instance, medical document collections).

Local feedback strategies are based on expanding the query with terms correlated to the query terms. Such correlated terms are those present in local clusters built from the local document set. Thus, before we discuss local query expansion, we discuss strategies for building local clusters. Three types of clusters are covered: association clusters, metric clusters, and scalar clusters.

Association Clusters

An association cluster is based on the co-occurrence of stems (or terms) inside documents. The idea is that stems which co-occur frequently inside documents have a synonymity association. Association clusters are generated as follows.

Definition *The frequency of a stem s_i in a document d_j, $d_j \in D_l$, is referred to as $f_{s_i,j}$. Let $\vec{m}=(m_{ij})$ be an association matrix with $|S_l|$ rows and $|D_l|$ columns, where $m_{ij}=f_{s_i,j}$. Let $\vec{m}^t$ be the transpose of $\vec{m}$. The matrix $\vec{s}=\vec{m}\vec{m}^t$ is a local stem-stem association matrix. Each element $s_{u,v}$ in $\vec{s}$ expresses a correlation $c_{u,v}$ between the stems s_u and s_v namely,*

$$c_{u,v} = \sum_{d_j \in D_l} f_{s_u,j} \times f_{s_v,j} \qquad \textbf{(5.5)}$$

The correlation factor $c_{u,v}$ quantifies the absolute frequencies of co-occurrence and is said to be unnormalized. Thus, if we adopt

$$s_{u,v} = c_{u,v} \qquad \textbf{(5.6)}$$

then the association matrix $\vec{s}$ is said to be unnormalized. An alternative is to normalize the correlation factor. For instance, if we adopt

$$s_{u,v} = \frac{c_{u,v}}{c_{u,u} + c_{v,v} - c_{u,v}} \qquad \textbf{(5.7)}$$

then the association matrix $\vec{s}$ is said to be normalized. The adoption of normalization yields quite distinct associations as discussed below.

Given a local association matrix $\vec{s}$, we can use it to build local association clusters as follows.

Definition *Consider the u-th row in the association matrix $\vec{s}$ (i.e., the row with all the associations for the stem s_u). Let $S_u(n)$ be a function which takes the u-th row and returns the set of n largest values $s_{u,v}$, where v varies over the set of local stems and $v \neq u$. Then $S_u(n)$ defines a local association cluster around the stem s_u. If $s_{u,v}$ is given by equation 5.6, the association cluster is said to be unnormalized. If $s_{u,v}$ is given by equation 5.7, the association cluster is said to be normalized.*

Given a query q, we are normally interested in finding clusters only for the $|q|$ query terms. ~~Further, it is desirable to keep the size of such clusters small.~~ This means that such clusters can be computed efficiently at query time.

Despite the fact that the above clustering procedure adopts stems, it can equally be applied to non-stemmed keywords. The procedure remains unchanged except for the usage of keywords instead of stems. Keyword-based local clustering is equally worthwhile trying because there is controversy over the advantages of using a stemmed vocabulary, as discussed in Chapter 7.

Metric Clusters

[illegible]

Association clusters are based on the frequency of co-occurrence of pairs of terms in documents and do not take into account *where* the terms occur in a document. Since two terms which occur in the same sentence seem more correlated than two terms which occur far apart in a document, it might be worthwhile to factor in the distance between two terms in the computation of their correlation factor. Metric clusters are based on this idea.

7

Definition *Let the distance $r(k_i, k_j)$ between two keywords k_i and k_j be given by the number of words between them in a same document. If k_i and k_j are in distinct documents we take $r(k_i, k_j) = \infty$. A local stem-stem metric correlation matrix $\vec{s}$ is defined as follows. Each element $s_{u,v}$ of $\vec{s}$ expresses a metric correlation $c_{u,v}$ between the stems s_u and s_v namely,*

$$c_{u,v} = \sum_{k_i \in V(s_u)} \sum_{k_j \in V(s_v)} \frac{1}{r(k_i, k_j)}$$

In this expression, as already defined, $V(s_u)$ and $V(s_v)$ indicate the sets of keywords which have s_u and s_v as their respective stems. Variations of the above expression for $c_{u,v}$ have been reported in the literature (such as $1/r^2(k_i, k_j)$) but the differences in experimental results are not remarkable.

The correlation factor $c_{u,v}$ quantifies absolute inverse distances and is said to be unnormalized. Thus, if we adopt

$$s_{u,v} = c_{u,v} \tag{5.8}$$

then the association matrix $\vec{s}$ is said to be unnormalized. An alternative is to normalize the correlation factor. For instance, if we adopt

$$s_{u,v} = \frac{c_{u,v}}{|V(s_u)| \times |V(s_v)|} \tag{5.9}$$

then the association matrix $\vec{s}$ is said to be normalized.

Given a local metric matrix $\vec{s}$, we can use it to build local metric clusters as follows.

Definition *Consider the u-th row in the metric correlation matrix $\vec{s}$ (i.e., the row with all the associations for the stem s_u). Let $S_u(n)$ be a function which* *takes the u-th row and returns the set of n largest values $s_{u,v}$, where v varies over the set of local stems and $v \neq u$. Then $S_u(n)$ defines a local metric cluster around the stem s_u. If $s_{u,v}$ is given by equation 5.8, the metric cluster is said to be unnormalized. If $s_{u,v}$ is given by equation 5.9, the metric cluster is said to be normalized.*

Scalar Clusters

One additional form of deriving a synonymity relationship between two local stems (or terms) s_u and s_v is by comparing the sets $S_u(n)$ and $S_v(n)$. The idea is that two stems with similar *neighborhoods* have some synonymity relationship. In this case we say that the relationship is indirect or induced by the neighborhood. One way of quantifying such neighborhood relationships is to arrange all correlation values $s_{u,i}$ in a vector $\vec{s}_u$, to arrange all correlation values $s_{v,i}$ in another vector $\vec{s}_v$, and to compare these vectors through a scalar measure. For instance, the cosine of the angle between the two vectors is a popular scalar similarity measure.

Definition *Let $\vec{s}_u = (s_{u,1}, s_{u,2}, \ldots, s_{u,n})$ and $\vec{s}_v = (s_{v,1}, s_{v,2}, \ldots, s_{v,n})$ be two vectors of correlation values for the stems s_u and s_v. Further, let $\vec{s} = (s_{u,v})$ be a scalar association matrix. Then, each $s_{u,v}$ can be defined as*

$$s_{u,v} = \frac{\vec{s}_u \cdot \vec{s}_v}{|\vec{s}_u| \times |\vec{s}_v|} \qquad \textbf{(5.10)}$$

The correlation matrix $\vec{s}$ is said to be induced by the neighborhood. Using it, a scalar cluster is then defined as follows.

Definition *Let $S_u(n)$ be a function which returns the set of n largest values $s_{u,v}$, $v \neq u$, defined according to equation 5.10. Then, $S_u(n)$ defines a scalar cluster around the stem s_u.*

Interactive Search Formulation

Stems (or terms) that belong to clusters associated to the query stems (or terms) can be used to expand the original query. Such stems are called neighbors (of the query stems) and are characterized as follows.

A stem s_u which belongs to a cluster (of size n) associated to another stem s_v (i.e., $s_u \in S_v(n)$) is said to be a *neighbor* of s_v. Sometimes, s_u is also called a *searchonym* of s_v but here we opt for using the terminology *neighbor*. While neighbor stems are said to have a synonymity relationship, they are not necessarily synonyms in the grammatical sense. Often, neighbor stems represent distinct keywords which are though correlated by the current query context. The local aspect of this correlation is reflected in the fact that the documents and stems considered in the correlation matrix are all local (i.e., $d_j \in D_l$, $s_u \in V_l$).