the obtained documents are given to the user. If the first number is larger, then all bound values are increased by some predetermined value (our incremental value is equal to 20). The matrix is transformed with the new bound values and the algorithm constructs a new query formulation. This query formulation is used for a new search, and the number of obtained documents is compared with the number requested by the user. If the numbers are equal, the output is given to the user; if the number of documents found during search is greater than requested, then the bound values are incremented again (using the same incremental value); and if the obtained number of documents is smaller than requested, then the incremental value is divided by 2 and a new value is used to decrease the bound values. Again the matrix is transformed, a new query formulation is constructed, and another search is performed. If the number of obtained documents is different from the required number, then the incremental value is divided by 2 and the result is added to the bound values if the difference between the obtained and required number of documents is positive or subtracted from the bound values if the difference is negative. This procedure is repeated until the numbers are equal or the incremental value is less than 1. (System developers who do not need such precision can choose any desirable approximation.) In both cases the output is given to the user and in the second case the output contains the number of documents closest to the required number.

This method of automatically computing bound values can be used even when the user does not tell the system the number of documents required in the output. In this case, the algorithm chooses a default (provided by the system's developers), for example, 10 documents.

This approach can be used to determine the likely order of importance of the documents for the user. When we increase the bound values, the number of descriptors in the subrequests is increased and we expect to obtain a higher precision level in the search. Now given the user's query (a marked set of documents) we can set bound value to obtain an output consisting of, say, 10 documents, and then decrease the bound values to obtain an output consisting of 20 documents. Then from assumptions of our algorithm we can conclude that the first 10 documents will be a subset in the second output and they will be more important (relevant) than the rest of the documents in the second output.

7.6

Some Aspects of Algorithm Functioning in Information Retrieval Systems

In this chapter we described an algorithm that a system can use to automatically construct a query formulation thereby replacing a human expert. The algorithm is designed to make a user's interaction with the IR system as simple as possible. It frees the user from many complicated and time-consuming operations required in constructing query formulations. It is clear that very little preparation is required from a user to successfully perform a search in an information retrieval system where the suggested algorithm is implemented. Such a system also simplifies the correction of query formulation on the basis of the search output. The user only needs to indicate which documents in the output are pertinent. These documents will be added to the marked set, and the algorithm will construct a modified (more precise) query formulation. However, as was mentioned earlier, the problem of correcting a query formulation is quite complex and for this reason will be considered further in Chapter 9.

In discussing when a specific set of descriptors will be considered as a subrequest, we mentioned that one of the criteria is the occurrence frequency of this set in the marked set of documents. These occurrence frequencies will be chosen automatically by the algorithm and will depend on the number of documents in the marked set.

In the experiment described earlier we obtained the occurrence frequencies by analyzing requests where the number of documents in the marked set (m.s.) varied from 1 to 12. In each of these cases, we compared the results of the searches for all possible occurrence frequencies. For example, when m.s. = 5, we tried occurrence frequencies of 1, 2, 3, 4, and 5 and compared the results of the search. In this case we found that an occurrence frequency (o.f.) of 2 gave the best results. Similarly we found the best occurrence frequencies for all m.s. with values from 1 to 12.

Based on our experiments we concluded that the best occurrence frequency is equal to k where $3^*(k - 1) < \text{m.s.} \leq 3^*k$. Hence, if $1 < \text{m.s.} \leq 3$, then the occurrence frequency is equal to 1 (we write o.f. = 1); for $4 \leq \text{m.s.} \leq 6$, o.f. = 2; for $7 \leq \text{m.s.} \leq 9$, o.f. = 3; and for $10 \leq \text{m.s.} \leq 12$, o.f. = 4. So this formula seems to provide a simple decision rule for choosing occurrence frequencies for a set of descriptors that will determine if this set of descriptors will be considered as a separate subrequest. Yet clearly the system developers can choose some other, possibly more sophisticated method for determining the best values of occurrence frequencies.

It is important to mention that the ability to vary the bound values and the occurrence frequencies of different sets of descriptors in the marked set of documents provides a necessary flexibility of the algorithm, which makes it adaptable to the requirements of different information retrieval systems.

Now we are going to take a look at another important feature of our algorithm. As we mentioned earlier, in the construction of a query formulation the main problem is to decide which descriptors should be included and how they should be combined in a query formulation. The result of this decision process is quite different for the manual construction and the proposed algorithm. In selecting descriptors for subrequests, a human being uses reasoning and chooses the descriptors that are most personally meaningful. However, it is not at all clear that a desirable query formulation should be constructed only