2. Restricting the output given to the user by finding the intersection of the outputs obtained during a search—by using different query formulations constructed by different algorithms—rather than by combining all these outputs. This not only reduces the size of the output but also increases the "pertinence" of every document in the output (because it was found by more than one algorithm). In this case, depending on the number of documents in the output and the number of algorithms, it is possible to vary the intersection from two to all of the obtained outputs.

3. Using algorithms for constructing query formulations (at least for the first step), which can restrict the output size and rank the documents in the output on the basis of their pertinence. Examples of such algorithms can be found in Chapter 7.

Notice that the first approach assumes the correction of the query formulation before the search is performed on the entire collection of documents. This correction is performed on the basis of the search characteristic $\sqrt{R \cdot P}$ described earlier. After the search is performed in a subcollection using combined query formulations, and the system obtains the user's evaluation of the pertinence of every document in the combined output, it is not only possible to select the best query formulation but also to compare all subrequests including subrequests from different query formulations. The correction may be done in several different ways. For example, we can select the best subrequests from all query formulations (obtained by different algorithms) used in the search and combine them into one query formulation, which will be used to search the entire collection. Another method would be to select the best algorithm, choose the best subrequests from the query formulation constructed by this algorithm, continue through several iteration steps on the same sample collection accumulating the best subrequests, and finally perform the search on the entire collection by using the combined query formulation.

The second and third approaches are suitable when the user explicitly bounds the number of documents in the output, for example, setting a limit of 20 documents. In this case, the third approach might be preferable because the ranking of documents provides a natural way to restrict the output. In the second approach the size of the output could still exceed the required bound.

9.10 Selective Feedback Algorithm for Dynamic Collection

The algorithm described next, which considers the choice of an optimal alternative for the search, is oriented toward a dynamic collection of documents. As in the case for a static collection, the user's evaluation of the output will be

used as input to this algorithm. In other words, the algorithm can proceed only when the user's evaluation is given to the system.

Before the initial search for a given search request, the query formulations are constructed by each of the available algorithms. The initial search is then conducted by each of the constructed query formulations and all outputs are combined. In other words, a combined output will be formed as a set union of all the outputs obtained by each query formulation. Because it is possible that the size of the combined output will be too large, and the user either cannot or will not want to survey it all, the combined output is ranked. Those documents that are found by more than one query formulation are considered more important (having more "weight") in the ranking; that is, the larger the number of query formulations "matching" the document, the higher its rank. Then the ranked combined output is given to the user for evaluation, and the user marks the pertinent documents. It is obvious that when it is impossible to survey and evaluate all the output, the user will survey and mark only a part of it, and the survey will start with documents with the highest rank. In this case, only the documents analyzed by the user are considered as the system's output. In other words, in this situation the user determines the number of documents in the output. For example, if the system found 170 documents but the user only inspects the 20 with the highest weight, then this situation is equivalent to the case in which the user requests the output of 20 documents and the system gives the user the 20 most important documents (see, for example, the method suggested by Salton, 1968).

At the first stage, the algorithm tests if there are any pertinent documents in the output. If there are none, the algorithm stops because there is no information for determining the choice of the system's best state. If pertinent documents do exist, the algorithm determines by which query formulation(s) they are found, and then further analysis is applied to those query formulations by which at least one pertinent document is found. The best query formulation (or their combination), from the point of view of the criterion r^2/n, indicates also the best algorithm that was used in its construction.

After determination of the best algorithm(s), the best subrequests are selected from all query formulations that are not among the best. This is done for the following reasons. First, it is possible that the subrequests that do not appear in the best query formulations give good search results and can prove to be effective for subsequent searches. Second, as was indicated earlier in the discussion of the character of feedback in a dynamic collection of documents, the best algorithm(s) will be used later for obtaining a new corrected query formulation, which means that there is no need to select the best subrequests from the best query formulations. Thus, for the evaluation of original subrequests in other than the best query formulations, the same criterion $\sqrt{R \cdot P}$ is used, and the best subrequests are considered to be those for which the value $\sqrt{R \cdot P}$ is *not lower* than value $\sqrt{R \cdot P}$ of the best query formulation.