to the descriptors depending on which iterations they obtained. They considered the descriptors used in the initial search more important. For this purpose they introduced "auxiliary query count (qcount) parameter designed to give extra weight to the original query terms." The use of qcount represents an artificial increase in the occurrence frequencies of the descriptors in the marked set of documents used for the automatic construction of query formulations.

The idea of qcount seems to be quite interesting although there are some questions as to the best way to use it. From our understanding of information need, in particular POIN, it seems more logical to use qcount for the descriptors appearing in the pertinent documents in the last output. This seems to follow from the changing boundaries of POIN attribute 4. The descriptors that were important for the search at the beginning may lose their significance, whereas new descriptors could better represent new boundaries of POIN. They will also be a part of the additional information necessary to construct a corrected query formulation. In any case, the use of qcount requires additional experimental support, and qcount could become a useful tool in better representing additional information about POIN. On the other hand, using qcount in static collections (as was done by the authors) does not seem to be as beneficial because there may be very little change in POIN.

In 1991 we published new, more complete algorithms of adaptive feedback both for static and dynamic collections of documents (Frants & Shapiro, 1991b). These algorithms formed the basis for the adaptive IR system developed at Fordham University. We describe these algorithms next, together with some experimental results. We start with the algorithm for the static collection of documents.

9.4 Feedback Algorithm for the Static Collection of Documents

We mentioned earlier that in realizing feedback for the static collection of documents, the task of control consists of an attempt to find pertinent documents, which were not found during the previous search, without an increase in the acceptable level of noise. A good illustration for better understanding this problem is presented in Figure 9.1. Here we discuss this figure with the goal of developing a feedback algorithm for the static collection of documents. The feedback algorithm in this situation will construct a new query formulation (area C), which will be used in an attempt to find new (presumably pertinent) documents. In other words, there is no reason to use in the new (corrected) query formulation any of the subrequests from the previous query formulation because such a subrequest can only find documents that form a subset of area B. Also, there is no reason to include in the output any documents found with the

new query formulation that were contained in area B. In developing an algorithm, these cases are taken into account.

The algorithm is applied after the system receives the user's evaluation of the previous output. The algorithm selects all documents marked as pertinent by the user (we denote the set of these documents by OM). If there are no such documents—that is, if OM is empty—then the algorithm stops, because there is no additional information about POIN given to the system. If OM is not empty, then the algorithm proceeds by combining sets OM and PM, where PM is the set used to obtain the query formulation in the previous search. In the first feedback cycle this set PM is the user's initial request and may consist of either one document (if the user's request is given in a sentence form) or several documents (if the user's request is a set of pertinent documents). This construction is described in more detail in Chapter 7.

The next step in the algorithm is the construction of a new query formulation using CM, which denotes the union of OM and PM, as input (recall that the set of documents used as an input for the construction of a query formulation is called a "marked set"). This step is a complex problem in its own right. In our case it is only one part of a more general feedback algorithm, and the query formulation obtained in this step is only a preliminary version (we denote it by AQ), which could undergo substantial changes. In solving the problem of constructing a query formulation, it is possible to use one of the existing algorithms or methods that accept as an input a marked set of documents or, as an alternative, to develop a completely new algorithm. For several reasons we suggest using the algorithm described in Chapter 7. First, the experimental results showed this algorithm to be quite effective. Second (in our opinion, a very important consideration), the algorithm allows us to construct query formulations without any limitation on the number of descriptors appearing in each subrequest (i.e., a subrequest may consist of either one descriptor or two, three, four, or more descriptors joined by the Boolean operator AND). Third, the algorithm allows for a simple control of the number of documents in the output; that is, when the user is interested in obtaining a specific number of documents in the output the algorithm can easily meet this requirement. In addition, at the request of the user the documents in the output can be ranked.

After obtaining AQ, the algorithm compares it to the combined query formulation (we denote it by CQ), which is obtained by combining all query formulations used in the previous feedback iterations during a search process in the same collection of documents. In the case of the first iteration, CQ coincides with the query formulation constructed from the initial request. Then the algorithm obtains a new query formulation (NQ) by removing from AQ all subrequests that appear in CQ and also those subrequests that contain any of the subrequests appearing in CQ. For example, if a, b are descriptors from the system's dictionary, and $a \wedge b$ is a subrequest of AQ, it will not appear in NQ if $a \wedge b$ is a subrequest of CQ or if either a or b is a subrequest of CQ. This is done