IV. The Cornell Programs for Cluster Searching and Relevance Feedback

E. Ide, R. Williamson, D. Williamson

1. Design Criteria

For the present write-up, it is assumed that the reader is familiar with the SMART project, originally implemented on an IBM 7094 at Harvard [1,2]. When this information retrieval system was rewritten for a CDC 1604 at Cornell, the programs available at Harvard could not be used directly. At that time, several design decisions for the Cornell system were made:

a. To continue to run the text analysis programs at Harvard [2], and to rewrite only those programs dealing with document abstracts and queries as numeric vectors;

b. to concentrate on two major areas at Cornell (the implementation of Rocchio's clustering algorithm [3] and of various relevance feedback algorithms [3,4]);

c. to write these programs to handle collections as large as the 1400 document, 225 query Cranfield collection;

d. to make the resulting system available for use and for experimental reprogramming to students on one-semester projects.

2. Basic Cornell System Organization

In order to implement the last of the points mentioned, the system was designed using many small routines. This modular structure adds a large degree of flexibility. An experimenter can generally achieve his purpose by modifying one or more small, self-contained subroutines rather than by having to rewrite a subsection of a large, complex program. The system subroutines communicate through COMMON blocks, and the common system variables are rigidly defined for the experimental programmer. This procedure minimizes errors and simplifies debugging. Further, it facilitates expansion of the system. Often a new program can be built around a set of basic routines already available in the system.

Most of the subroutines are written in Fortran rather than in basic machine language. The sacrifice of speed over assembly coding is slight with the fast Fortran compiler available on the Control Data 1604. This decision permits an easier modification of the code and makes conversion of the system to other computers more reasonable.

The large document collection with which the system was eventually to be used (criterion C) precludes the internal core storage of all the document vectors simultaneously, under any packing scheme. Thus, it is necessary to put the documents onto a tape to be read serially, with advance buffering to overlap computation and tape movement as much as possible. Since it is desirable to use preliminary and cross-check runs on various different collections, it was felt that it would be easiest if all document collections were to reside on one system tape, rather than on many different tapes. In addition, production runs referencing more than one collection would be much more convenient to run, since only one tape would be needed.

3. System Implementation of Rocchio's Clustering Algorithm

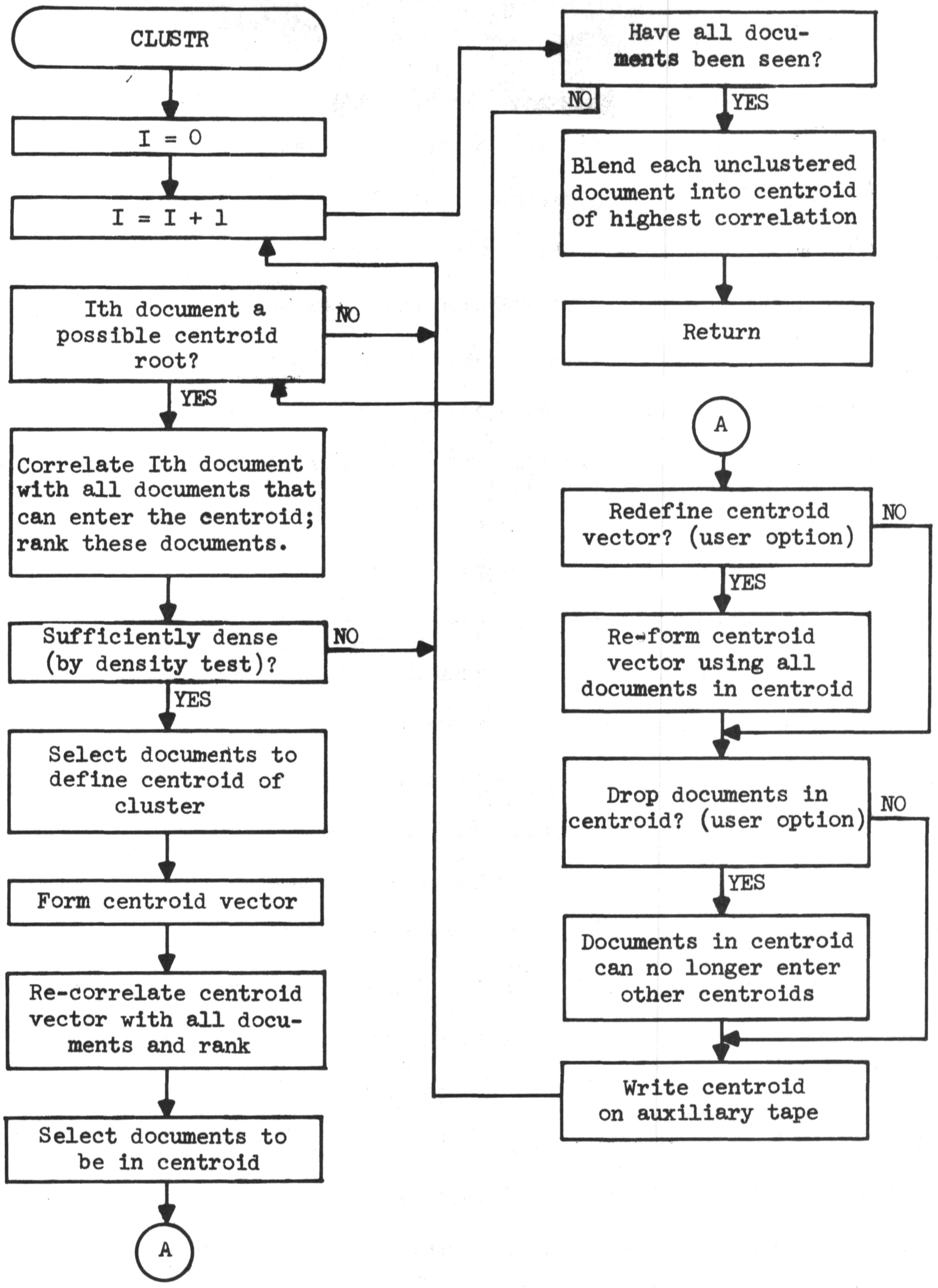
In future libraries, collections of fifty to one-hundred-thousand documents are contemplated. For collections of this size, the linear search schemes used for testing purposes would take an unmanageably long time, even with batch-processed queries. If any direct user interaction were desired, the time cost of a linear search would be prohibitive in dollars, and the user probably would not be content to wait for up to thirty minutes, the time required to perform a fast linear search on fifty-thousand vectors.

A clustering system hopefully can group similar documents together, and replace the group of documents usually correlated by a single vector representing the general area covered by those documents. If one-hundred-thousand documents could be usefully grouped into one-thousand groups of two-thousand documents each (it is often desirable to place a document in more than one group), only about three-thousand comparisons, as opposed to one-hundred-thousand, would be needed to find most of the documents relevant to a given query. This reduction of search time and money is desirable, and hence considerable efforts are being made to generate effective clustering algorithms.

A clustering algorithm is useful only if the resulting document groups permit a more efficient search, while maintaining the same quality of results provided by a full search of the document collection. Thus, there are two separate problems associated with clustering algorithms; that of clustering the collection into groups, or "centroids", and that of evaluating the quality of the search results using these centroids. Each of these problems is handled by a separate system subroutine. The first subroutine, CLUSTR, groups the documents, and the second, CENVAL, evaluates these groupings, or evaluates any other centroid groupings previously calculated.

Subroutine CLUSTR forms clusters of documents for two-level searches using Rocchio's algorithm. [3] The routine successively considers each document in a collection as a possible root of a centroid. It then correlates this document with all other documents in the collection, and applies Rocchio's density test to the resulting ranked documents. If the root passes the density test, it is considered to be a centroid root, and the documents which will define the centroid are chosen from the top of the ranked list. A "centroid vector" is then formed, which is the center in the document space consisting of the defining documents. All documents are re-correlated with this centroid vector and additional documents to be included in the centroid are chosen from the top of the new ranked list. A user option allows the centroid vector to remain unchanged or to be redefined at this time to include all documents in the centroid. Another option determines whether these documents now included in the centroid are dropped, or not dropped, from the document space. The program then picks a second possible root and repeats the above process, until all possible centroid roots have been tried. If, at this point, there are documents that are not included in any centroid, they may be inserted into the cluster whose centroid root is most highly correlated with them. In this manner, CLUSTR places all documents in groups represented by their centroid, as seen in Fig. 1. Parameters for Rocchio's density test, number and size of clusters desired, and several other experimental variables are controlled by the system user.

Subroutine CENVAL performs up to four searches of the document collection. One of these is a standard full search for comparison purposes. The other three runs use the centroids, as generated by a CLUSTR run, for the given collection. To search the collection using the centroids, a query


CLUSTR
I = 0
I = I + 1
Ith document a possible centroid root?
NO
YES
Correlate Ith document with all documents that can enter the centroid; rank these documents.
Sufficiently dense (by density test)?
NO
YES
Select documents to define centroid of cluster
Form centroid vector
Re-correlate centroid vector with all documents and rank
Select documents to be in centroid
A
Have all documents been seen?
NO
YES
Blend each unclustered document into centroid of highest correlation
Return
A
Redefine centroid vector? (user option)
NO
YES
Re-form centroid vector using all documents in centroid
Drop documents in centroid? (user option)
NO
YES
Documents in centroid can no longer enter other centroids
Write centroid on auxiliary tape

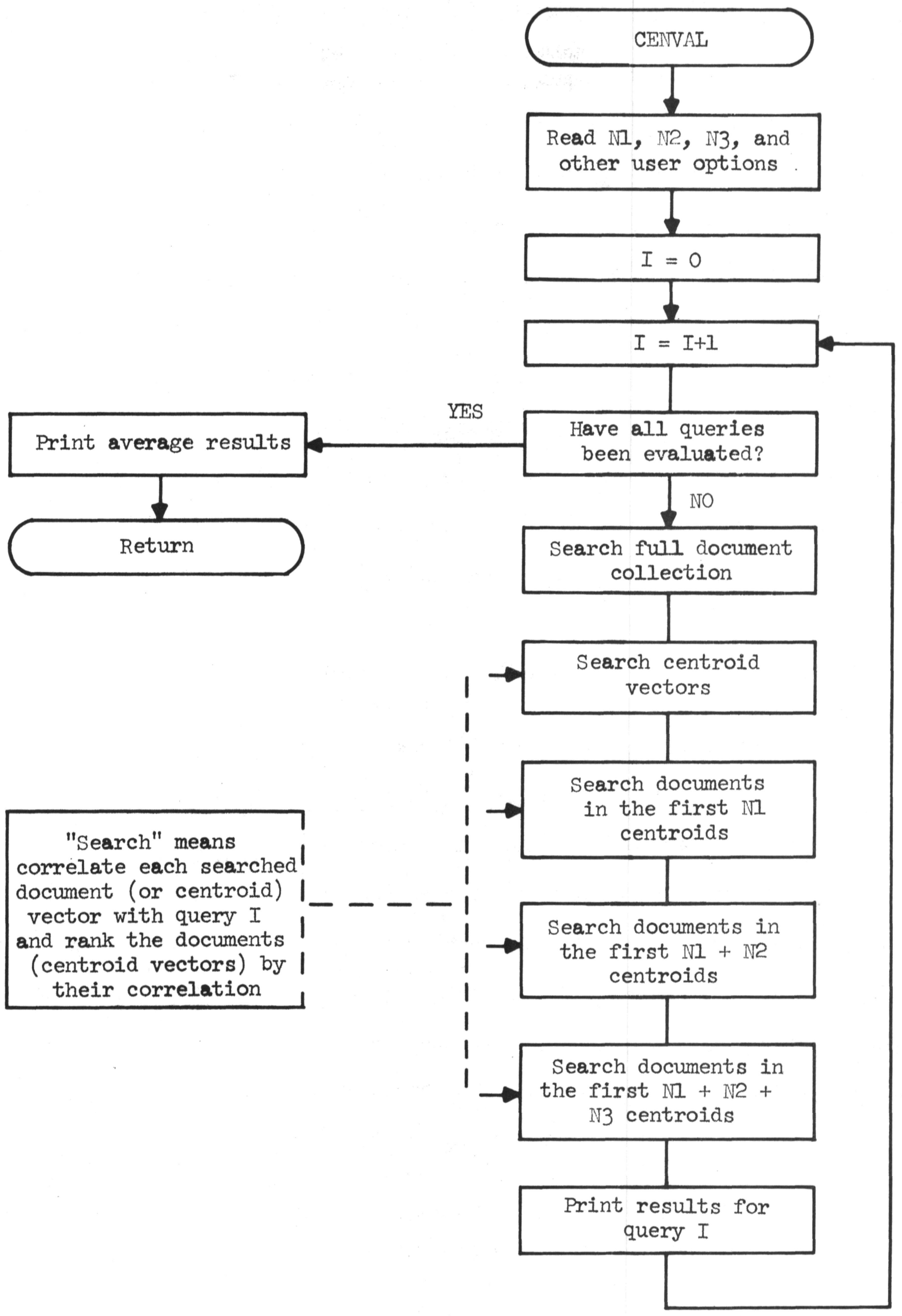
Flowchart for Routine CLUSTR

Fig. 1

is first correlated with each centroid, and the centroids ranked according to this correlation. Then for each of the three centroid searches, a user-supplied number of these centroids are pulled off the top of the ranked list, and the documents included in these centroids are ranked and listed. The performance measure used to evaluate the "quality" of these searches is the recall-precision graph normally used by the SMART system [5]. A recall-precision graph is plotted for each query, and average statistics are given for the set of queries. A flowchart for the CENVAL routine is shown in Fig. 2.

4. System Implementation of Relevance Feedback

Relevande feedback is a process whereby a poor query can be improved using feedback supplied by a user after he is shown a small fraction of the documents in the system. The user first submits an initial draft of his query, which is processed by the system. The user is then given a few documents that the system believes to be most relevant to his initial request. The user indicates which of these documents are relevant, and which are non-relevant to his needs. The system uses this information to modify the original query, and searches again. This process can be iterated as often as desired. The relevance feedback process is often justified by noting that if the query is <u>not</u> stated in terms similar to those used for most of the relevant documents, the system may, nonetheless, be able to locate one or two of the relevant documents. These documents may be more highly correlated with the remaining relevant documents than the initial query, and may therefore provide valuable input for a query modification procedure.


CENVAL
Read N1, N2, N3, and other user options
I = 0
I = I+1
Have all queries been evaluated?
YES
NO
Print average results
Return
Search full document collection
Search centroid vectors
Search documents in the first N1 centroids
Search documents in the first N1 + N2 centroids
Search documents in the first N1 + N2 + N3 centroids
Print results for query I
"Search" means correlate each searched document (or centroid) vector with query I and rank the documents (centroid vectors) by their correlation

Flowchart for Routine CENVAL

Fig. 2

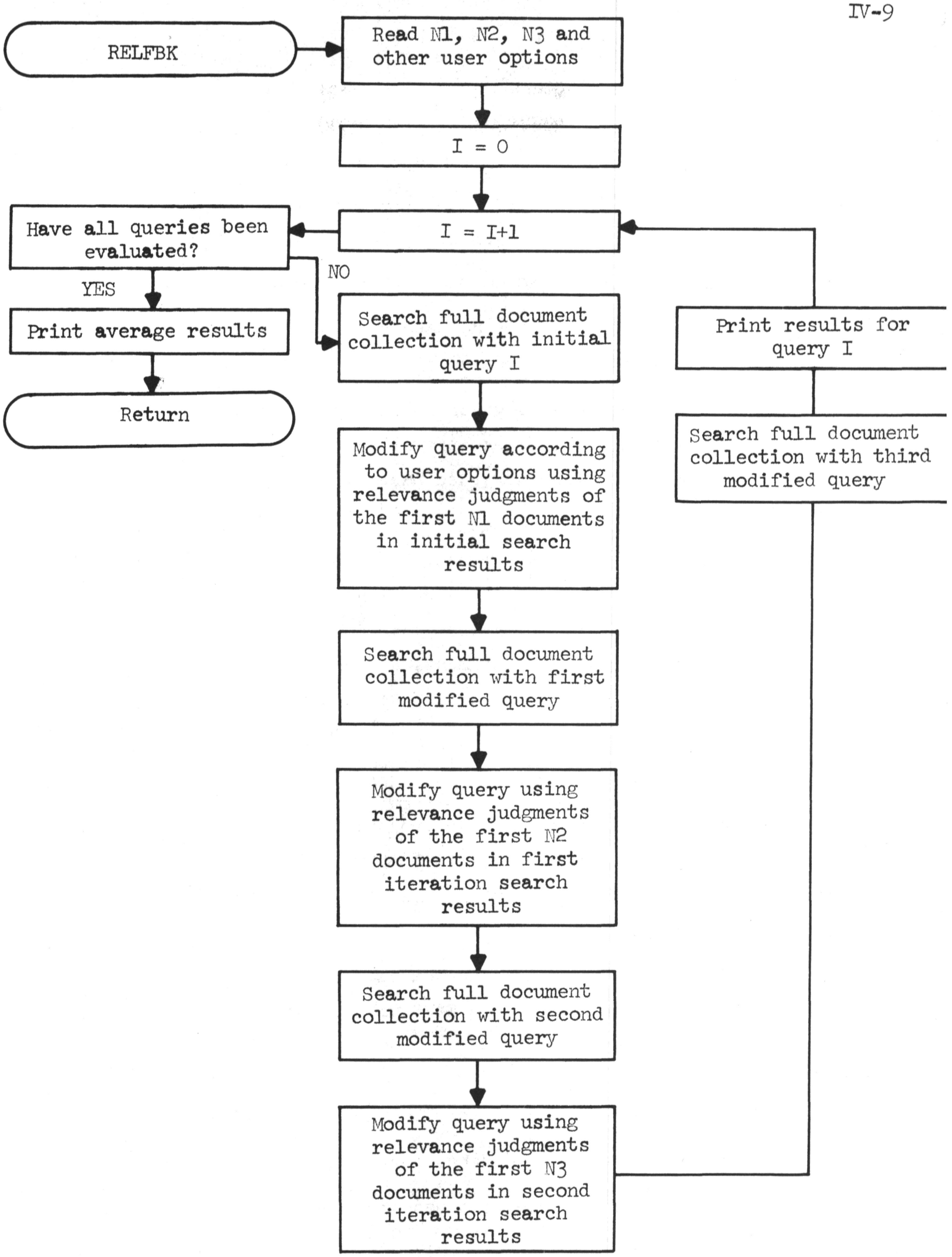
The relevance feedback algorithm described is implemented in the Cornell system by subroutine RELFBK. RELFBK first performs an initial search of the collection. Three feedback iterations, each altering the query, are then performed for each query. A feedback iteration first alters the query by combining the weights in a given number of the documents most highly correlated with the query, and then performs a new complete search. By means of control cards, the experimenter can add to the query those concepts present in the relevant documents, or subtract those concepts which are present in the non-relevant documents. The number of documents recovered on each iteration (that is, the number that might be used to update the query) can also be varied. (See Fig. 3)

5. Further Details of the Cornell System Organization

A) Routines Available Without User Reprogramming

A basic control routine was written to permit execution of some standard jobs. This routine, named EXEC, can be used in most cases without modifications to execute most possible sequences of desired procedures. The EXEC routine reads control cards, each containing the alphabetic name of one of the system subroutines. Upon reading a control card, EXEC calls the named subroutine, which executes and then returns control to EXEC so that another control card can be read. The basic system jobs listed below can be executed with control cards through the EXEC routine:

a. START — initializes a new document tape with no collections. This routine is used before making a new collection tape;


RELFBK
Read N1, N2, N3 and other user options
I = 0
I = I+1
Have all queries been evaluated?
YES
NO
Print average results
Return
Search full document collection with initial query I
Modify query according to user options using relevance judgments of the first N1 documents in initial search results
Search full document collection with first modified query
Modify query using relevance judgments of the first N2 documents in first iteration search results
Search full document collection with second modified query
Modify query using relevance judgments of the first N3 documents in second iteration search results
Search full document collection with third modified query
Print results for query I

Flowchart for Routine RELFBK

Fig. 3

b. RESTRT - initiates the use of a document tape which does contain collections. Either START or RESTRT must be the first control card in each computer run;

c. CLUSTR - groups the documents (or queries) in a given collection using Rocchio's clustering algorithm (see part 3);

d. CENVAL - searches the documents in a given collection using document clusters created by CLUSTR. (see part 3);

e. RELFBK - searches a given collection of documents using a given set of queries, providing up to three iterations of relevance feedback (see part 4);

f. MAKTAP - reads a document (or query) collection from cards and adds it to the document tape; MAKTAP also provides and stores on the document tape summary statistics for the collection, including the maximum concept number and maximum weight of each document, and the maximum number of concepts in any document in the collection;

g. LISTAP - lists a given document or query collection from the document tape;

h. TIME (I) - prints the time recorded on the internal computer clock numbered I; if I is negative, TIME sets clock $|I|$ to zero. This routine is used to obtain the execution time of any section of the run.

i. MASTER - an undefined system name to be used to call a user-written subroutine from EXEC without modifying the existing system subroutines.

B) Service Routines Available to the Experimental Programmer

Several input-output and other service routines are available to the experimental programmer. These routines are also used by the basic subroutines listed in section A above. All routines in section A can also be called by user-written routines.

A set of special subroutines, listed below, is used to handle all movements of the document tape (DOCTAP). These routines provide buffering and are efficient in tape movement. Further, if the experimental programmer uses these routines exclusively for all DOCTAP movements, the document tape is protected from unintentional change by his routines.

a. START, RESTRT, MAKTAP, LISTAP. (see section A)

b. NEWCOL - locates a given document (or query) collection on DOCTAP, positioning the tape to read the first document of the collection; it also reads and stores in COMMON the first record of the collection, which contains collection parameters used by other subroutines;

c. LOCDOC - reads a given document within the collection that was selected by the last previous call to NEWCOL;

d. DOCMOV - a fast basic machine language subroutine that moves DOCTAP forward or backward over a specified number of end-of-files, which serve to demarcate collection boundaries. It is used by the other subroutines that move DOCTAP.

A set of anuxiliary tape subroutines is available to be used for temporary storage of data on tapes other than DOCTAP.

a. WRITE (VEC) and READ (VEC) - single-channel double buffered tape reading and writing routines;

b. CRDCEN - generates a centroid tape from the documents, specified on cards, defining a centroid (document group) vector, and from the documents contained in that centroid. This routine makes it possible to reuse the clusters resulting from a run of CLUSTR.

c. CENDOC - transfers a set of document vectors from an auxiliary tape to DOCTAP, and defines these vectors as a new document collection; this routine is used to store the centroid vectors generated by CLUSTR on DOCTAP, to be used later for centroid clustering or for re-evaluation of different two-level search schemes;

d. DOCCEN - transfers a collection from DOCTAP to an auxiliary tape; this routine is used to reverse CENDOC so that stored centroid vectors may be reused.

Other important service routines available are:

a. INNER - forms the inner product (correlation) of two documents in core storage. Any one of three different correlation coefficients can be used;

b. SRTUP and SRTDWN - sorts a system vector (called CORR) in numeric (SRTUP) or inverse numeric (SRTDWN) order. These sort routines are high speed routines and use n log n locations in CORR, where n is the number of items to be sorted;

c. MKITAP - correlates each document in a centroid (read from a tape of centroid vectors) with the centroid itself, and graphs the distribution of documents within a centroid;

In addition, a system of associated printer-plotter routines is available which can be used to print output graphs with user-controlled scale and plotting characters.

References

[1] G. Salton et al., Report No. ISR-9 to the National Science Foundation, Harvard Computation Laboratory, Cambridge, Mass., August, 1965.

[2] G. Salton et al., Report No. ISR-11 to the National Science Foundation, Department of Computer Science, Cornell University, Ithaca, New York, June, 1966.

[3] J. J. Rocchio, Jr., Document Retrieval Systems — Optimization and Evaluation, Report No. ISR-10 to the National Science Foundation, Harvard Computation Laboratory, Cambridge, Mass., March, 1966.

[4] W. Riddle, T. Horwitz, and R. Dietz, Relevance Feedback in Information Retrieval Systems, Report No. ISR-11 to the National Science Foundation, Section VI, Computer Science Department, Cornell University, Ithaca, New York, June, 1966.

[5] G. Salton, the Evaluation of Automatic Retrieval Procedures — Selected Test Results Using the SMART System, American Documentation, Vol. 16, No. 3, July, 1965, pp. 209-222.