XII. Query Splitting in Relevance Feedback Systems

A. Borodin, L. Kerr and F. Lewis

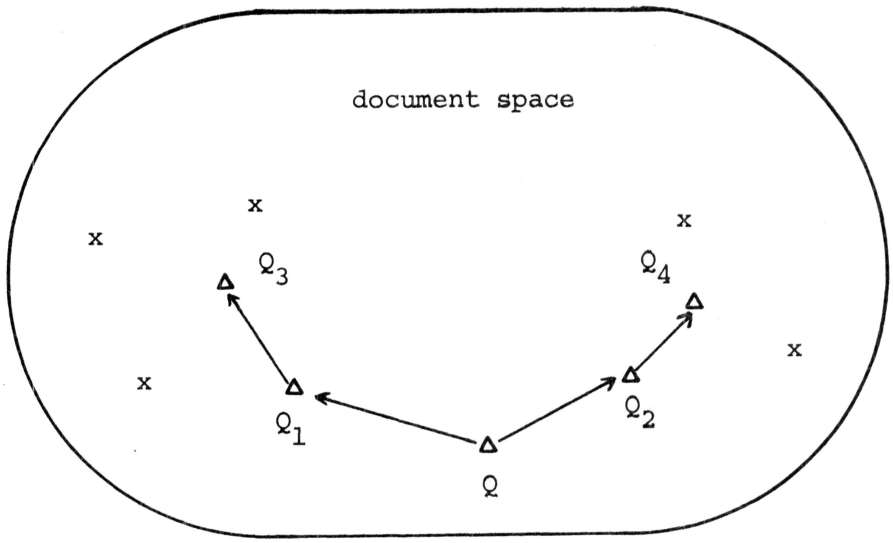
Abstract

A modification of normal relevance feedback is presented. It is suggested that instead of simply modifying a query on each iteration, two or more new queries may be formed. Evaluation of experimental results shows this method produces some improvement. A new measure is introduced which permits extrapolation of these results to large document collections. With regard to this measure, query splitting appears to be best suited for large systems.

1. Introduction

Query splitting is an extension of the standard relevance feedback procedure. Instead of simply modifying a query on each iteration, two or more new queries may be formed. Such a procedure often provides improved results. Consider the example shown in Fig. 1.

Q is the original query. If a simple feedback algorithm moves the query toward one of the groups of relevant documents, then the relevant documents in the other group will not be retrieved. In fact, the query may not move significantly to either group. Ideally, one would like to replace Q with two queries, Q_1 and Q_2. The user then makes relevance judgments on the documents retrieved by Q_1 and Q_2, and a relevance feedback algorithm can be applied to Q_1 and Q_2 separately, producing new queries Q_3 and Q_4.



x represents relevant documents

Δ represents queries

Query Splitting Example

Fig. 1

Query splitting is beneficial when the relevant documents are located in distinct groups in the document space. Such a grouping is not merely hypothetical. User requests may be general and deal with more than one topic. Moreover, the structure of the document space is based on predetermined correlation judgments which do not necessarily reflect the user's conception of relevance. Query splitting is not always required, of course, and indiscriminate use of the method may lead to inefficiencies. Part of the query splitting strategy, therefore, is to decide when a query should be split.

2. The Query Splitting Algorithm

As mentioned above, query splitting is embedded within a relevance feedback system. The system first reads in the user's initial query, Q_o. New queries are then formed iteratively, as follows:

1. The documents in the collection are ranked according to their correlation with Q_i.
2. The five highest ranking documents, not previously retrieved, are presented to the user for relevance judgments.
3. The user indicates to the system which of the retrieved documents were relevant.
4. The system examines these relevant documents to see if they form distinct groups. This decision is based on the document-document correlations of the relevant documents, relative to the average correlation between Q_i and the first five documents retrieved

by Q_i. If a document-document correlation does not exceed a constant (TP) times the average query-document correlation, then the two documents are considered to belong to separate groups. In this way, the system clusters the relevant documents into zero (if no relevant documents are retrieved), one, or several groups.

5. For each group j a new query Q_{i+1}^{j} is formed according to the relevance feedback formula

$$Q_{i+1}^{j} = Q_i + \sum_k r_k - \sum_k n_k$$

where $\{r_k\}$ represents the relevant documents in the group and $\{n_k\}$ represents the two highest ranking nonrelevant documents retrieved by Q_i. If Q_i retrieved no new relevant documents, then just one new query is formed, using the negative feedback formula

$$Q_{i+1} = Q_i - \sum_k n_k$$

where $\{n_k\}$ again represents the two top nonrelevant documents.

6. The above steps are then repeated for each of the new queries separately, with the exception that in step 2 only three documents are retrieved from each split query. This avoids a proliferation of retrieved documents when query splitting occurs.

The query splitting algorithm used in the experimental analysis was not as specific as the one presented above. It allowed for more extensive query splitting as well as a more general feedback formula. A description of the program is given in the Appendix. The algorithm described above was derived by using the strategy which gave the best results after trying a number of different parameter values.

3. Evaluation and Results

In order to implement and test the query splitting strategy, a program was written to simulate the feedback portion of the SMART system. The Cranfield thesaurus collection of 200 documents and 42 queries was used to compare various strategies. Final evaluation is provided by a comparison of the performance of relevance feedback with and without the query splitting strategy.

Twenty-four of the forty-two queries in the Cranfield collection produced some query splitting (i.e. two or more relevant documents were retrieved on some iteration). Evaluation is restricted to these twenty-four queries, since for the other queries, regular feedback and query splitting perform identically. The following methods were compared:

1. Regular feedback.
2. Query splitting with the threshold parameter TP = 1.5.
3. Query splitting with TP = 0.75 (results in fewer splits).

One possible measure of performance is a "user measure" in which recall and precision values are determined from the order in which the user receives the documents. Conventions for such ordering are not difficult to establish. Table 1 below shows the improvement over regular feedback in the number of relevant documents retrieved, as a function of the number of retrieved documents. Only the queries for which such differences appear are listed. These results tend to favor query splitting, especially for larger numbers of retrieved documents.

Query No.	Increase in no. of rel. doc's retrieved									
	TP = 1.5					TP = 0.75				
	No. of doc's retrieved					No. of doc's retrieved				
	5	10	15	20	25	5	10	15	20	25
3	-	-	-	-	+1	-	-	-	-	-
5	-	-1	-1	-1	-1	-	-	-	-	-
7	-	+1	+2	+2	+2	-	+1	+2	+2	+2
13	-	-1	-	-	-	-	-	-	-	-
15	-	-	-2	-	-	-	-	-1	-1	-
16	-	-	-1	-	-	-	-	-	-	-
28	-	-	-1	-	-	-	-	-	-	-
31	-	-	+1	+1	+1	-	-	-	-	-
35	-	-1	-	-	-	-	-	-	-	-
36	-	-1	-	-	-	-	-1	-	-	-
38	-	+1	-	-	-	-	-	-	-	-
Total Improvement	0	-2	-2	+2	+3	0	0	+1	+1	+2

The User Measure

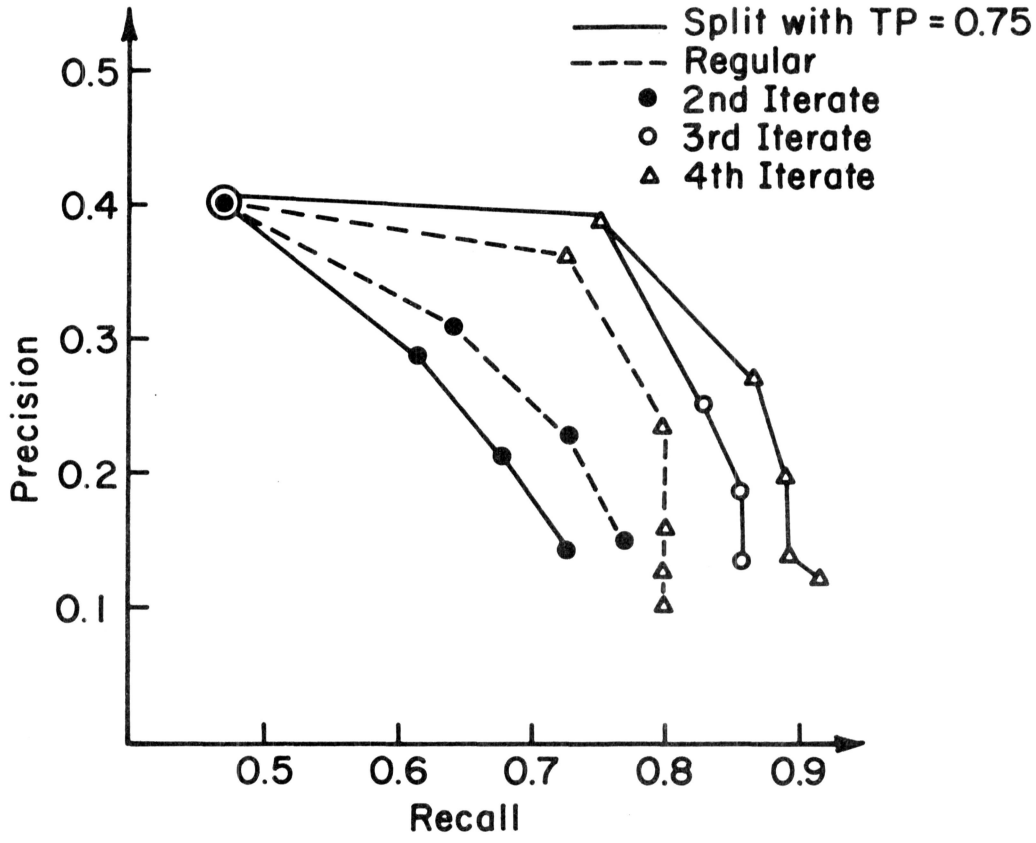
Table 1

Another measure, better suited for evaluating overall system performance, is obtained as follows: recall and precision values are computed for the ranking of documents in each iteration, considering previously retrieved documents to be removed from the document space. The recall-precision curves averaged over 24 search requests are shown in Figs. 2 and 3. Once again, general improvement for 0.75 query splitting is illustrated, while improvement for 1.5 query splitting is restricted to the higher recall range.

There exist two reasons for believing that the previous results tend to be meaningful:

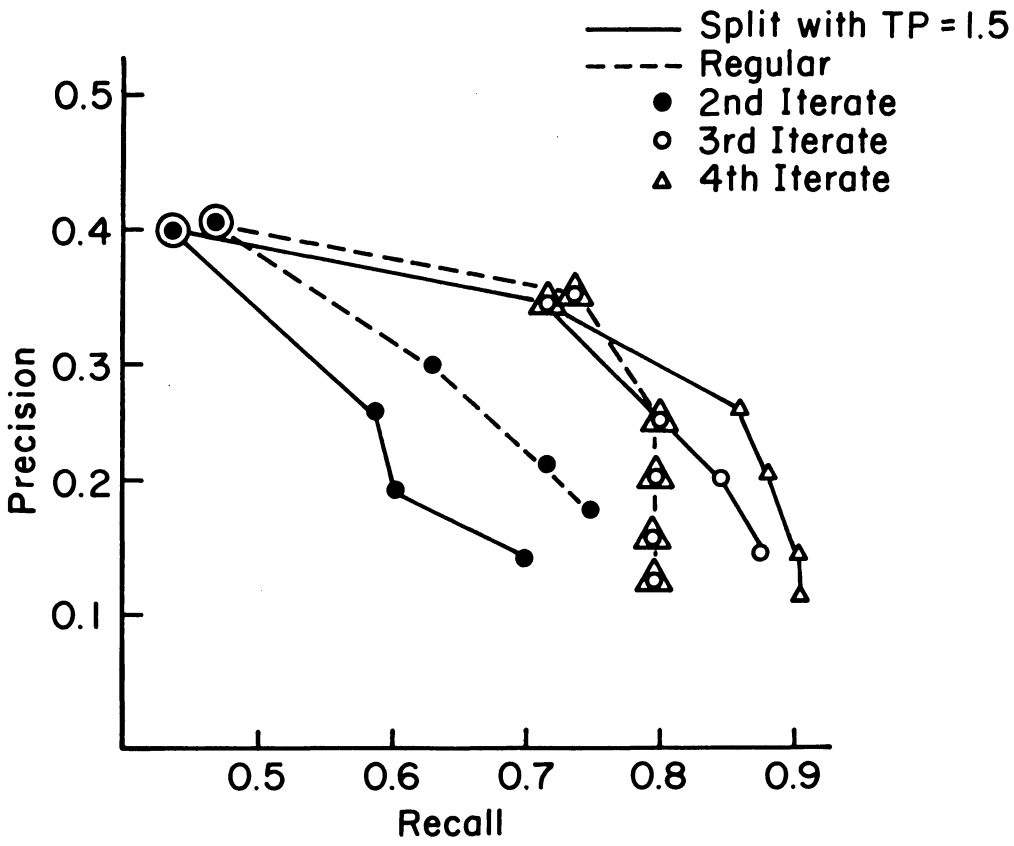
1. The Cranfield 200 is a small homogeneous collection which provides neither the diversification of document space nor the user population which leads to the type of general request mentioned in the introduction.
2. The measures used, especially the user measure, are too "sensitive" to the large number of documents retrieved in proportion to the size of the document space.

Fig. 4 illustrates how sensitive Query 7 is to the number of retrieved documents per iteration. Gross differences in recall and precision occur depending on whether five or ten documents are retrieved after the first iteration. Moreover, Fig. 4 shows that Document 95 would never be retrieved with regular feedback procedures in a large collection if the document rankings were kept proportionate. This is in contrast to the performance of query splitting illustrated in Fig. 4b. The document-document correlations for the relevant documents of Query 7 are shown in Fig. 5.



Query Splitting (TP = 0.75) vs. Regular Feedback
(averages over 24 queries)

Fig. 2



Query Splitting (TP = 1.5) vs. Regular Feedback
(averages over 24 queries)

Fig. 3

Rank	Iteration Number			
	1	2	3	4
1	41*R	41 R	100	90 R
2	100*	90 R	127*	42*R
3	90*R	156*	187*	11
4	111*	91*	41 R	41 R
5	11*	96*	196*	199
6	45	199*	24*	156
7	110	29*	128*	188*
8	127	60	72 R	45*
9	104	23	103	111
10	192	109	39	100
11	71	72 R	26	29
12	159	95 R	17	173*
13	42 R	193	170	39*
14	76	42 R	99	104
15	133	56	84	192
16	185	155	154	71
17	176	11	104	159
18	83	188	158	176
19	196	141	83	184
20	156	184	69	76
35	72 R	0	0	0
38	0	0	0	72 R
44	95 R	0	0	0
50	0	0	95 R	0
66	0	0	0	95 R
92	0	0	42 R	0
130	0	0	90 R	0

a) Regular Feedback

Rank	Iteration	Split Subqueries	
	1	2	3
1	41*R	41 R	90 R
2	100*	100	91*
3	90*R	71*	11
4	111*	111	111
5	11*	39*	95*R
6	45	83*	93*
7	110	25	110
8	127	84	94
9	104	110	192
10	192	29	195
11	71	155	159
12	159	127	109
13	42 R	45	104
14	76	156	100
15	133	92	76
16	185	114	96
17	176	153	121
18	83	23	199
19	196	192	176
20	156	90 R	82
24	0	72 R	0
25	0	0	0
29	0	0	41 R
35	72 R	0	0
37	0	95 R	42 R
44	95 R	0	0
59	0	42 R	0
60	0	0	72 R
79	0	0	0
86	0	0	0
89	0	0	0
102	0	0	0
137	0	0	0

b) Query Splitting (TP = 0.75)

*Indicates Retrieved Document

Sensitivity of Number of Retrieved Documents per Search Iteration

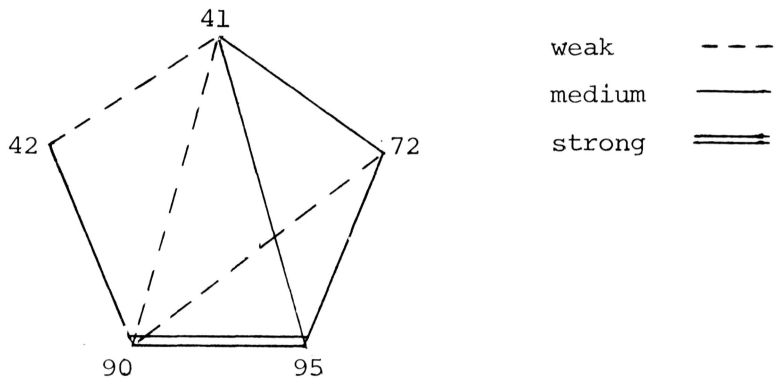
Fig. 4

A new measure is introduced which attempts to capture the notion of "relative improvement" between iterations. Consider a relevant document with initial rank R. Suppose one iteration of relevance feedback causes the document to attain a new rank of R/3. Then if subsequent iterations sustain this "rate of convergence", the number of iterations required to retrieve the document is the least i which satisfied

$$(1/3)^{i} R \leq n$$

where n is the number of documents given to the user. For this example, the rate of convergence r is 1/3.

In the case of query splitting, the descendents become independent queries, each with its own rate of convergence for any previously unseen relevant document. The best of these rates is taken to be the true rate of convergence. This is justifiable, since query splitting is designed to iterate towards individual groups, and the document need only be retrieved by one of the split queries. However, to be consistent with regular feedback, a document is considered to be retrieved only when its rank becomes less than n/S where S is the number of queries into which the original query was split.



Query 7
Correlations Between Relevant Documents of Fig. 4
Fig. 5

Formally, let $\hat{r}$ be the expected rate of convergence, and assume that $\hat{r}$ is maintained throughout all iterations. This assumption may not be completely valid; however, it was found, experimentally, that $\hat{r}$ is maintained at least as well for query splitting as it is with regular feedback. A document with initial rank R will be retrieved within i iterations if

$$(\hat{r})^{i} \cdot R \leq n/\hat{S}$$

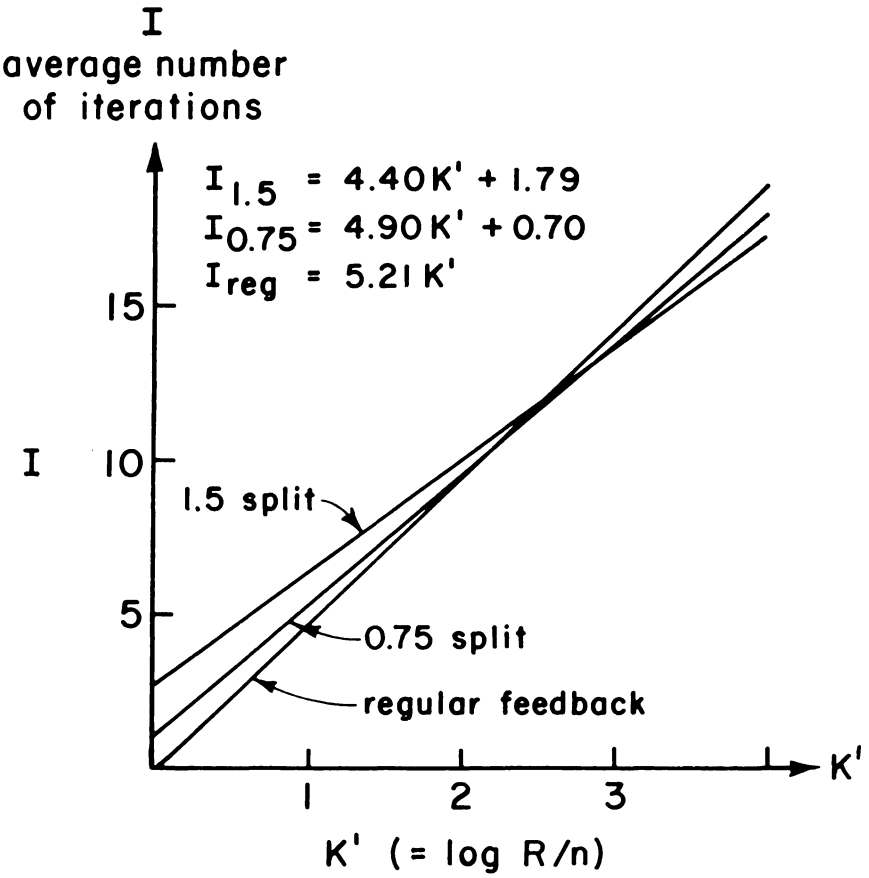
where $\hat{S}$ is the expected number of queries generated in the splitting process. Algebraic manipulation yields

$$i \geq \frac{\log (R/n) + \log \hat{S}}{- \log \hat{r}}$$

Since $\hat{r} < 1$, $\log \hat{r}$ is negative. This motivates defining a rate of improvement $\hat{M} = - \log \hat{r}$.

From the experimental results, values of $\hat{r}$ and $\hat{S}$ were obtained for each query, for the three methods. Then i was computed in terms of a parameter $K' = \log (R/n)$. These values of i were averaged over the seventeen queries of the original twenty-four which did not retrieve all relevant documents on the first iteration, and the average , I, was plotted against K' for each method. The results are shown in Fig. 6.

Since I (the average number of iterations) is an indication of retrieval effort, the best method is the one which yields minimal values of I. The curves of Fig. 6 show that query splitting is not beneficial unless $K' > 2.2$, i.e. $R/n > 150$. Since R is the same order of magnitude as the size



Expected Iterations Required for Relevant Document Retrieval

Fig. 6

of the document collection, this means that in large collections, query splitting would probably produce better results than those obtained with the Cranfield 200, for which $K' \approx 1$.

4. Conclusions and Suggestions for Further Research

As in all systems, one must weigh expected gains against the cost of a proposed improvement. Although the query splitting algorithm is easy to implement, the cost of the increased computing time may prove prohibitive. However, the algorithm can be readily modified for use with clustered document spaces. In this form, it should be realistically considered for use with the large collections for which query splitting appears to be most appropriate.

The following topics are suggested for further research:

1. Some scheme for dropping nonproductive queries would result in improved precision and a reduction in computing time.
2. A potentially useful modification would consist in generating by negative feedback an additional query on each iteration. Such a strategy might retrieve documents which could not be retrieved in any other way. These queries would have a higher probability of being nonproductive, however, so this modification should be implemented in conjunction with suggestion 1.
3. A better understanding of the topology of relevant documents in the document space might lead to more complex but more efficient query splitting strategies.

References

[1] S. Friedman, J. Maceyak, S. Weiss, A Relevance Feedback System Based on Document Transformation, Information Storage and Retrieval, Report ISR-12 to the National Science Foundation, Section X, Department of Computer Science, Cornell University, June 1967.

[2] H. Hall and N. Weiderman, The Evaluation Problem in Relevance Feedback Systems, Information Storage and Retrieval, Report ISR-12 to the National Science Foundation, Section XII, Department of Computer Science, Cornell University, June 1967.

[3] E. Ide, User Interaction with an Automated Information Retrieval System, Information Storage and Retrieval, Report ISR-12 to the National Science Foundation, Section VIII, Department of Computer Science, Cornell University, June 1967.

[4] J. Kelly, Negative Response Relevance Feedback, Information Storage and Retrieval, Report ISR-12 to the National Science Foundation, Section IX, Department of Computer Science, Cornell University, June 1967.

[5] G. Salton, Search and Retrieval Experiments in Real-Time Information Retrieval, Technical Report 68-8, Department of Computer Science, Cornell University, February 1968.

Appendix

A Brief Introduction to the FALTER System

1. Introduction

The FALTER system was developed as a programming device used to test the feasibility of query splitting. In order to thoroughly check this idea, a general purpose experimental system was required.

Flexibility, simplicity, and modularity are maintained throughout the system. It consists of ten independent subroutines controlled by a main program. The interfaces between the routines are exceedingly simple so that modifications and substitutions can be made easily.

Every aspect of the relevance feedback portion is input controlled. Also, the amount of output can be indicated through input parameters. These features will be described below.

2. General Algorithm

The FALTER system is built on a skeleton of relevance feedback, producing new queries according to the formula:

$$Q_{i+1} = \pi Q_i + \omega Q_o + \alpha \sum_{lna}^{na} r_j + \mu \sum_{lnb}^{nb} n_k \quad .$$

A method of query splitting has been added to the standard relevance feedback mechanism. This is done by first clustering the relevant documents retrieved by the query using the correlation coefficient for this purpose. The above formula is applied to produce the new queries taking one cluster at a time.

3. System Operation

The main portions of the system are indicated by the flow chart of Fig. 7. Some of the options in the system are described below:

a) All of the constants (ALPHA, MU, NA, etc.) are read in by the program.

b) The numbers of iterations desired, documents retrieved, queries processed are read in also.

c) Splitting is controlled by input parameters.

d) Three levels of output are provided as a trace feature. These levels are selected by input.

A listing of the routines follows:

1. Main Program

This routine controls operation, input and output.

Variables involved:

PI, ALPHA, MU, OMEGA, NA, LNA, NB, LNB are feedback constants

NUMIT - number of iterations

IRUN - number of queries

NUM - # docs retrieved/iteration

HINUM - # docs retrieved without split

LONUM - # docs retrieved/query on split

NNUM - length of output-chart

ISW =
- 0 positive feedback only
- 1 standard split - no negative
- 2 regular feedback - no split

ITRACE =
- -1 final output
- 0 slight trace
- 1 much output

TP - controls split (set to 0.75)

2. NEWQRY

This routine reads in information about a query.

TITLE - the query itself

NUMSEQ - its sequence # (1,2,3,etc.)

NUMCOL - its "collection #"

CONCPT - the concepts

WGHT - their weights

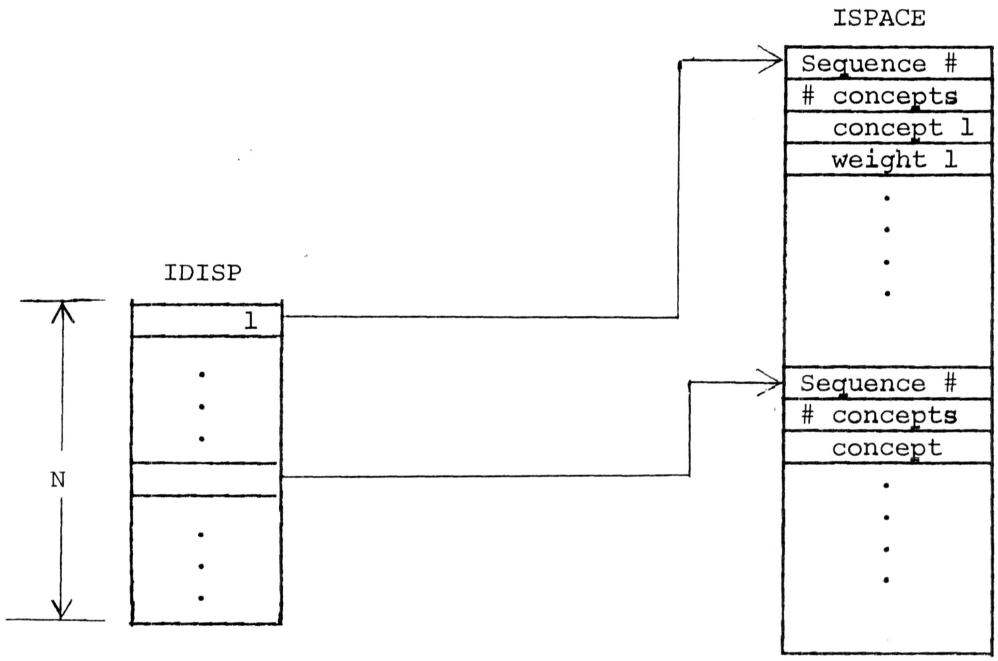
SUMSG - sum of weights squared

NUMREL - # of relevant documents

RELDOC - the relevant documents

3. INDOCS

This routine reads the documents in to a structure as below:



These are the document space and display table

SOCSQ - sum of weights squared for documents

4. CORLAT

A document-document correlation according to:

$$\frac{\sum d_i \, d'_i}{\sqrt{\sum d_i^2 \sum d_i'^2}}$$

5. COSINE

A query document correlator according to:

$$\frac{\sum d_i \cdot q_i}{\sqrt{\sum d_i^2 \sum q_i^2}}$$

The result is a list of the document numbers:
DOCNUM and the correlations: CORR

6. SORT

Simple $n^2/2$ sort on CORR and DOCNUM

7. PUTOUT

Output chart LIST made up from DOCNUM with correlations in ACORR and relevance indication in ISREL

8. JJREL

Checks the relevance of a document and returns an R where applicable

9. FEDBAK

   Performs vector modification and constructs new query

10. UANLIZ

    The relevant and irrelevant documents retrieved are indicated in IREL and JREL and they are noted in GOTIT

11. CONSOL

    Sorts part of output array

12. STRTGY

    The splitting routine which determines the query structure for this iteration using clustering operation

13. REGFDB

    Set up constants

14. CONST

    Set up constants

15. EVAL

    Executive routine for system which merges split queries into a composite query, and computes recall and precision values