DOCUMENT RESUME

ED 101 718 IR 001 570

AUTHOR	Salton, Gerard
TITLE	Information Storage and Retrieval Scientific Report No. ISR-22.
INSTITUTION	Cornell Univ., Ithaca, N.Y. Dept. of Computer Science.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	Nov 74
NOTE	380p.
EDRS PRICE	MF-$0.76 HC-$19.67 PLUS POSTAGE
DESCRIPTORS	*Algorithms; *Computer Programs; Computer Science; Content Analysis; Data Bases; Data Processing; Dictionaries; Feedback; *Indexing; Information Processing; *Information Retrieval; *Information Storage; Information Systems; Information Theory; Item Analysis; Thesauri
IDENTIFIERS	*File Structures

ABSTRACT

The twenty-second in a series, this report describes research in information organization and retrieval conducted by the Department of Computer Science at Cornell University. The report covers work carried out during the period summer 1972 through summer 1974 and is divided into four parts: indexing theory, automatic content analysis, feedback searching, and dynamic file management. Twelve individual papers are presented. (Author/DGC)

DEPARTMENT OF COMPUTER SCIENCE

CORNELL UNIVERSITY

INFORMATION STORAGE AND RETRIEVAL

Scientific Report No. ISR-22

to

The National Science Foundation

Ithaca, New York
November 1974

Gerard Salton
Project Director

Department of Computer Science

Cornell University

Information Storage and Retrieval

Scientific Report No. ISR-22

to

The National Science Foundation

Ithaca, New York
November 1974.

Gerard Salton
Project Director

TABLE OF CONTENTS

PART ONE

INDEXING THEORY

TABLE OF CONTENTS (continued)

PART TWO

CONTENT ANALYSIS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

PART THREE

FEEDBACK SEARCHING

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

PART FOUR

DYNAMIC DOCUMENT SPACE

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

Summary

The present report is the twenty-second in a series describing research in information organization and retrieval conducted by the Department of Computer Science at Cornell University. The report covering work carried out for approximately two years (summer 1972 to summer 1974) is divided into four parts: indexing theory (sections I to III), automatic content analysis (sections IV to VI), feedback searching (sections VII to IX), and dynamic file management (sections X to XII).

The normal schedule in the distribution of ISR reports has not been maintained in recent years, due largely to the scarcity of publication funds. For the same reason, a number of recently published articles covering related research work are not being reprinted in the present report. Interested readers may want to refer to the following additional items in particular:

a) Contributions to the Theory of Indexing (G. Salton, C.S. Yang, and C.T. Yu), Proc. IFIP Congress 74, North Holland Publishing Company, Amsterdam, 1974.

b) On the Specification of Term Values in Automatic Indexing (G. Salton and C.S. Yang), Journal of Documentation, Vol. 29, No. 4, December 1973, p. 351-372.

c) Proposals for a Dynamic Library (G. Salton), Information - Part 2, Vol. 2, No. 3, 1973, p. 5-25.

d) Theory of Indexing and Classification (C.T. Yu), Doctoral Thesis, Cornell University, Technical Report 73-181, Department of Computer Science, Cornell University, Ithaca, N.Y., August 1973, 238 pages.

Some time has been devoted during the last year to the design of an on-line implementation of the experimental SMART retrieval system, and test runs of the on-line version have been made on the IBM 370/168 computer at Cornell. The off-line version of the system continues to be used for experiments at various locations in the United States and abroad.

In recent years, increasing attention has been paid to the study of a variety of file organization and retrieval algorithms, including some that have not yet found their way into operational implementation. Among these is the use of clustered file manipulations instead of inverted directory searches, vector matching processes instead of keyword coincidence counting, dynamic document space modification, automatic file retirement procedures, and interactive retrieval methodologies.

The present report thus includes studies dealing with feedback searching and dynamic file modification. A great deal of emphasis has also been placed on the generation of new indexing theories which assign specific functions in content analysis to various indicators such as single terms, phrases, and thesaurus categories. These theories are explained in Part I of the present report.

Sections I and II by G. Salton, A. Wong, and C.S. Yang, and by A. Wong, respectively, cover investigations relating the density of the document space to the retrieval effectiveness obtainable with such a space. In particular, the earlier work dealing with the determination of term discrimination values makes it appear that "good" terms — those indicative of information content — are those which increase the dissimilarity between documents, that is, which spread out the document space. The experimental output in section I and II confirms that a low-density space is associated with effective retrieval, and vice-versa. Similarly, a high-density space provudes poor retrieval performance.

The theory in sections I and II is developed further in section III by G. Salton, C.S. Yang, and C.T. Yu relating the discrimination value of a term to its document frequency in a collection. The best discriminators are terms with medium document frequency. This fact is used to construct an optimum indexing vocabulary by turning high frequency single terms into phrases thereby reducing the document frequency, and assembling low frequency terms into thesaurus groups thus increasing the frequency. The effectiveness of the resulting indexing vocabulary is assessed by citing appropriate experimental evidence.

Sections IV to VI, constituting Part 2 of this report, deal with various aspects of automatic content analysis. Section IV by R. Crawford covers the construction and effectiveness of a variety of negative dictionaries ("stop lists") containing terms that should not be used for content identification. This work leads to the generation of an indexing vocabulary of optimum size. Section V by A. v.d. Meulen covers the operations of the so-called dynamic information values. In that system all term weights are fixed initially at some given value (say 1). Good terms, that is, those contained in useful documents are then increased in weight dynamically in the course of the operations. Bad terms are similarly demoted by reducing the term weights. The last section, number VI, by K. Welles deals with experiments leading to the construction of optimum term classifications (thesauruses) using the pseudo-classification method. This process utilizes a classification criteron based on user relevance assessments to group the terms rather than on the more usual semantic term similarities.

The next three sections, VII to IX, constitute Part 3 of this report, entitled feedback searching. Section VII by A. Wong, R. Peck, and A. v.d. Meulen attempts to determine relationships between the effectiveness of the initial content analysis (indexing) and the usefulness of iterative feedback searching. It is found that differences in the effectiveness of the initial indexing are preserved during the feedback operations. Section VIII by K. Sardana relates the length of the feedback query to the effectiveness of the retrieval operation. It is found that shorter feedback queries provide better retrieval; methods are therefore given for reducing feedback query length. A similar reduction in vector length is investigated by M. Kaplan in section IX, applied to the centroid vectors (profiles) representing the document groups in a clustered file organization.

Part 4, consisting of sections X to XII covers dynamic document space modification procedures. In section X by C.S. Yang the term discrimination values are used as parameters in the construction of appropriate document space modification methods. A document "utility value", determined by earlier user-system interactions, is similarly used for document space modification in section XI by A. Wong and A. v.d. Meulen. Finally, in section XII by K. Sardana a variety of automatic document retirement methods can be used automatically to reduce the size of the collection by eliminating items exhibiting low usefulness. Three retirement methods based respectively on average term weight measurements, document space modification methods, and the storage of special usage indicators are examined and their effectiveness is evaluated.

All earlier ISR reports in this series are obtainable from the National Technical Information Service in Springfield, Virginia. The order numbers for the last few reports are PB 214-020 (ISR-21), PB 211-061 (ISR-20), PB 204-946 (ISR-19) and PB 198-069 (ISR-18), respectively.

G. Salton

A Vector Space Model for Automatic Indexing

G. Salton, A. Wong, and C.S. Yang[+]

Abstract

In a document retrieval, or other pattern matching environment where stored entities (documents) are compared with each other, or with incoming patterns (search requests), it appears that the best indexing (property) space is one where each entity lies as far away from the others as possible; that is, retrieval performance correlates inversely with space density. This result is used to choose an optimum indexing vocabulary for a collection of documents. Typical evaluation results are shown demonstrating the usefulness of the model.

1. Document Space Configurations

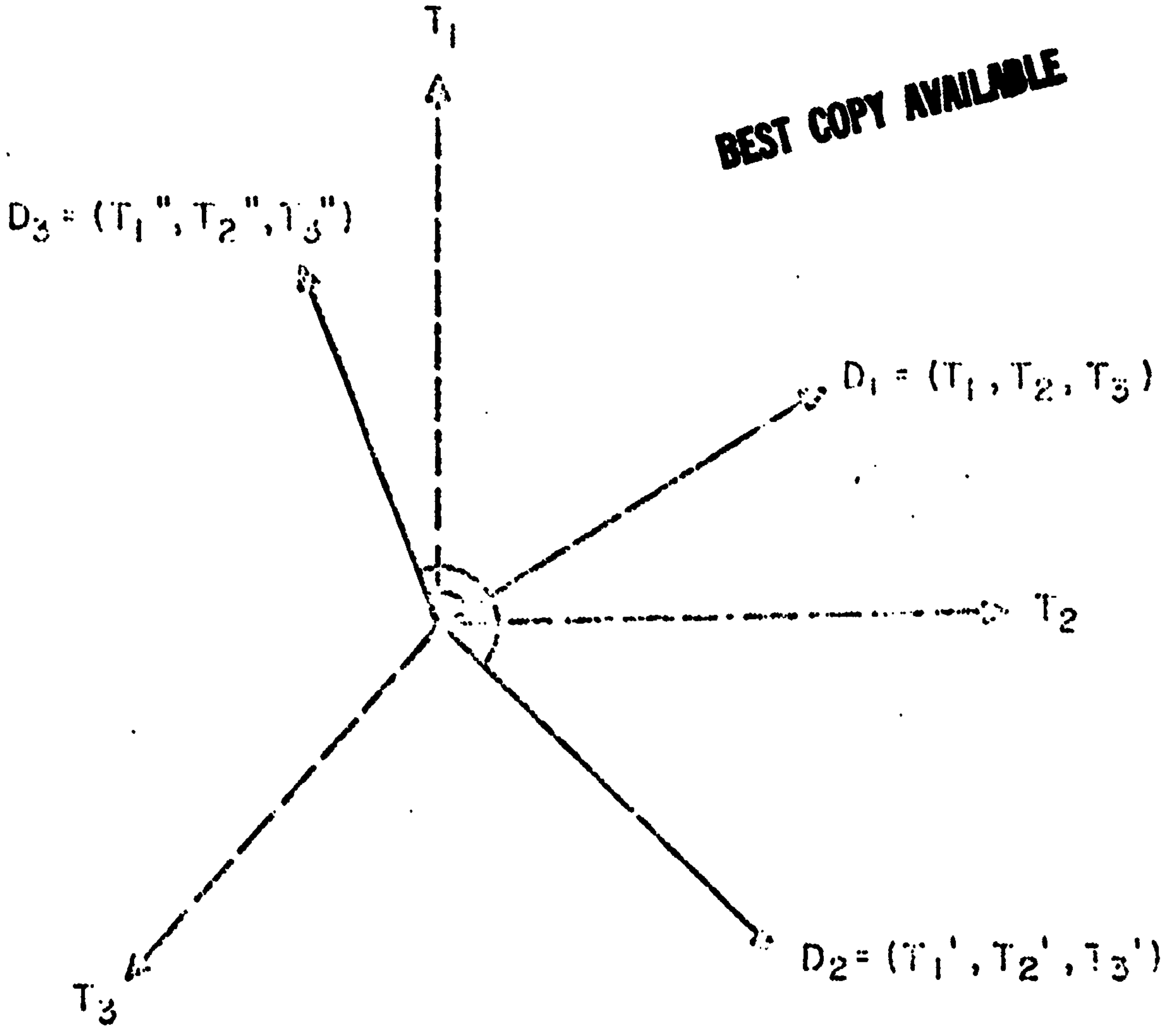
Consider a document space, consisting of documents D_i, each identified by one or more index terms T_j; the terms may be weighted according to their importance, or unweighted with weights restricted to 0 and 1.* A typical three-dimensional index space is shown in Fig. 1, where each item is identified by up to three distinct terms. The three dimensional example may be extended to t dimensions when t different index terms are present. In that case, each document D_i is represented by a t-dimensional vector

$$D_i = (d_{i1}, d_{i2}, \ldots, d_{it}),$$

d_{ij} representing the weight of the jth term.

[+]Department of Computer Science, Cornell University, Ithaca, N.Y., 14853

*Although we speak of documents and index terms, the present development applies to any set of entities identified by weighted property vectors.



Vector Representation of Document Space

Fig. 1

Given the index vectors for two documents, it is possible to compute a similarity coefficient between them $s(D_i, D_j)$, reflecting the degree of similarity in the corresponding terms and term weights. Such a similarity measure might be an inverse function of the angle between the corresponding vector pairs — when the term assignment for two vectors is identical, the angle will be zero producing a maximum similarity measure.

Instead of representing each document by a complete vector originating at the 0-point in the coordinate system, the relative position of the vectors is preserved by considering only the envelope of the space. In that case, each document is graphically identified by a single point whose position is specified by the area where the corresponding document vector touches the envelope of the space. Two documents with similar index terms are then represented by points that are very close together in the space: obviously the distance between two document points in the space is inversely correlated with the similarity between the corresponding vectors.

Since the configuration of the document space is a function of the manner in which terms and term weights are assigned to the various documents of a collection, one may ask whether an optimum document space configuration exists, that is, one which produces an optimum retrieval performance.*

If nothing special is known about the documents under consideration, one might conjecture that an ideal document space is one where documents that are jointly relevant to certain user queries are clustered together, thus insuring that they would be retrievable jointly in response to the corresponding

*Retrieval performance is often measured by parameters such as recall and precision, reflecting the ratio of relevant items actually retri ed, and of retrieved items actually relevant. The question concerning optimum space configurations may then be more conventionally expressed in terms of the relationship between document indexing on the one hand, and retrieval performance on the other.

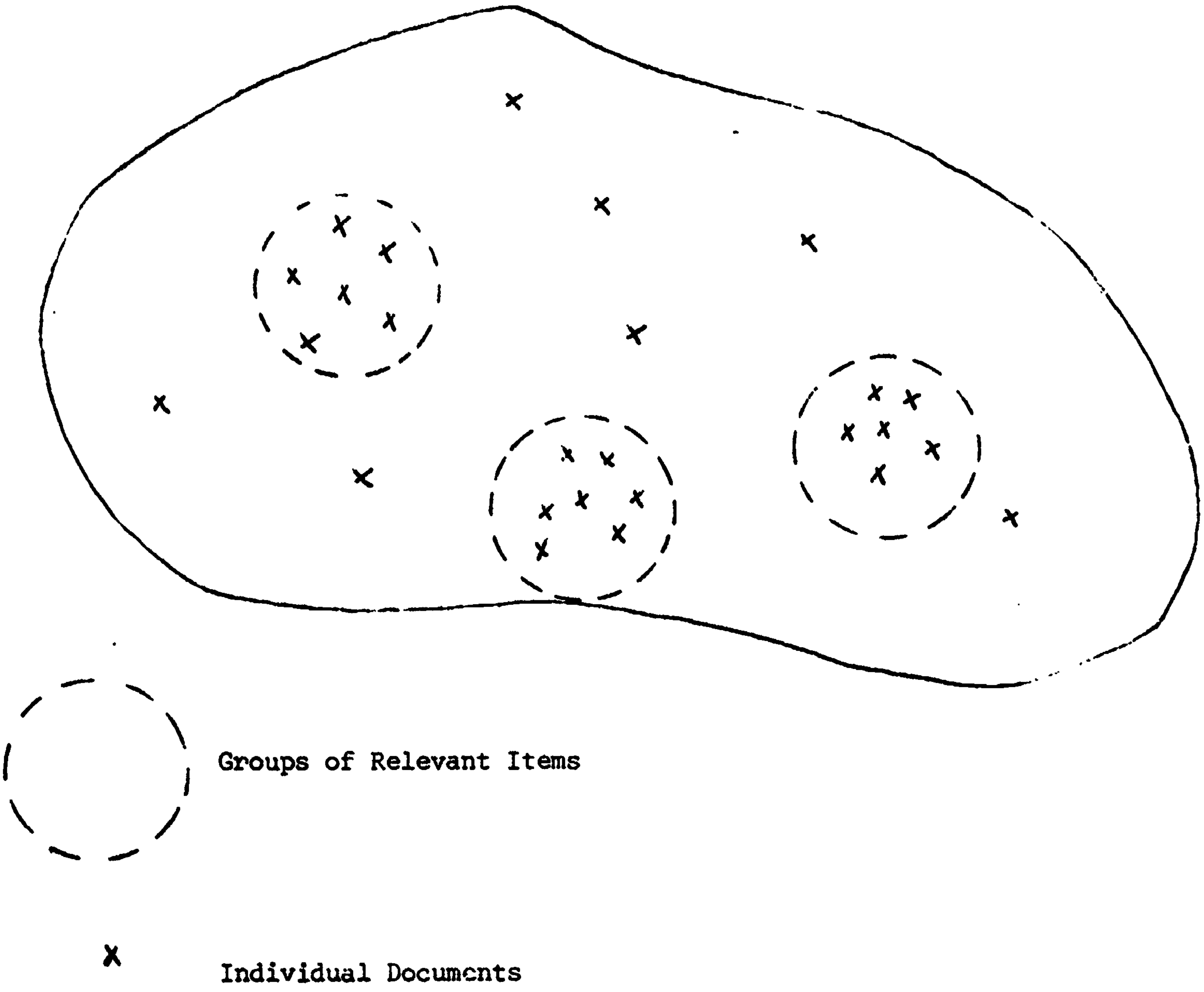
queries. Contrariwise, documents that are never wanted simultaneously would appear well separated in the document space. Such a situation is depicted in the illustration of Fig. 2, where the distance between two x's representing two documents is inversely related to the similarity between the corresponding index vectors.

While the document configuration of Fig. 2 may indeed represent the best possible situation, assuming that relevant and nonrelevant items with respect to the various queries are separable as shown, no practical way exists for actually producing such a space, because during the indexing process, it is difficult to anticipate what relevance assessments the user population will provide over the course of time. That is, the optimum configuration is difficult to generate in the absence of a priori knowledge of the complete retrieval history for the given collection.

In these circumstances, one might conjecture that the next best thing is to achieve a maximum possible separation between the individual documents in the space, as shown in the example of Fig. 3. Specifically, for a collection of n documents, one would want to minimize the function.

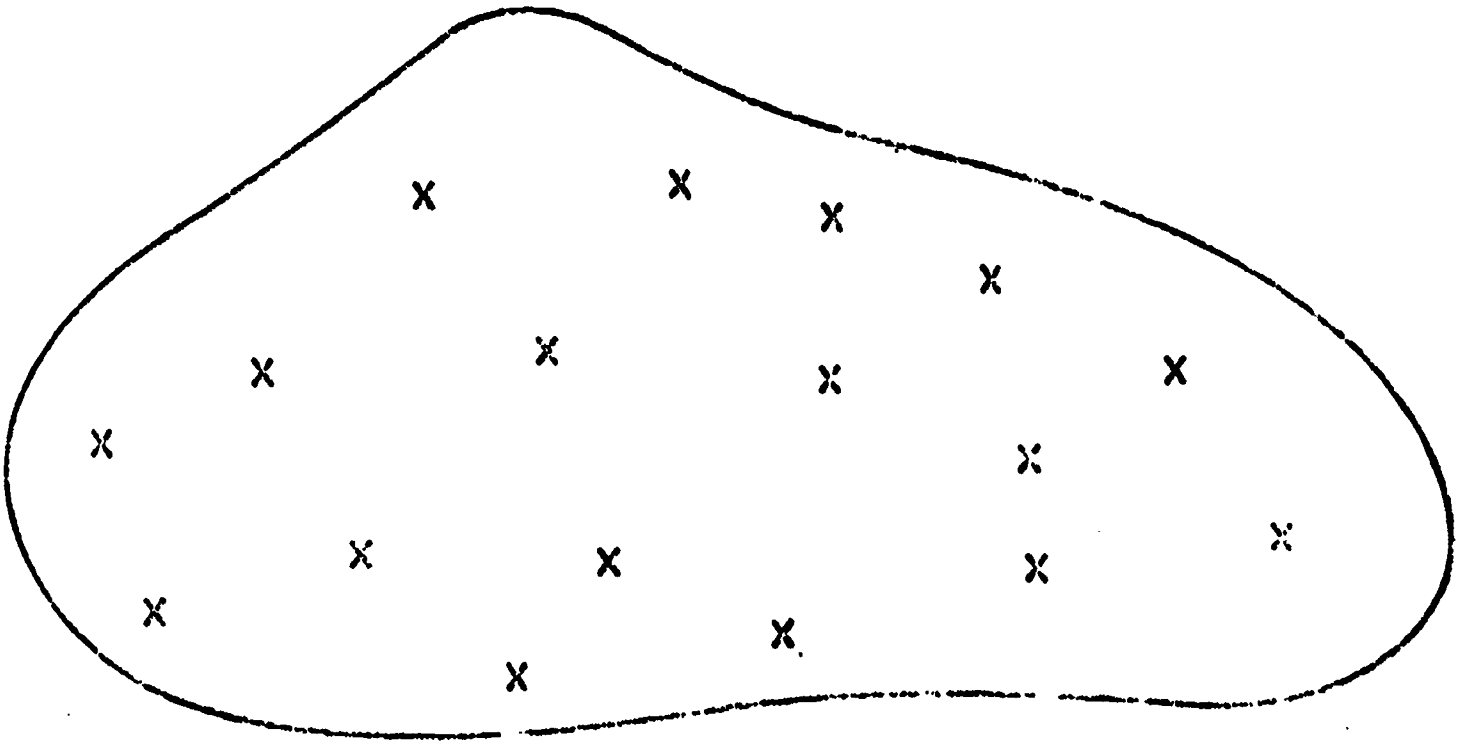
$$F = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} s(D_i, D_j), \qquad (1)$$

where $s(D_i, D_j)$ is the similarity between documents i and j. Obviously when the function of equation (1) is minimized, the average similarity between document pairs is smallest, thus guaranteeing that each given document may be retrieved when located sufficiently close to a user query <u>without also necessarily retrieving its neighbors</u>. This insures a high precision search output, since a given relevant item is then retrievable without also retrieving a number of nonrelevant items in its vicinity. In cases where several different



Ideal Document Space

Fig. 2



X Individual Document

Space with Maximum Separation

Between Document Pairs

Fig. 3

relevant items for a given query are located in the same general area of the space, it may then also be possible to retrieve many of the relevant while rejecting most of the nonrelevant. This produces both high recall and high precision.*

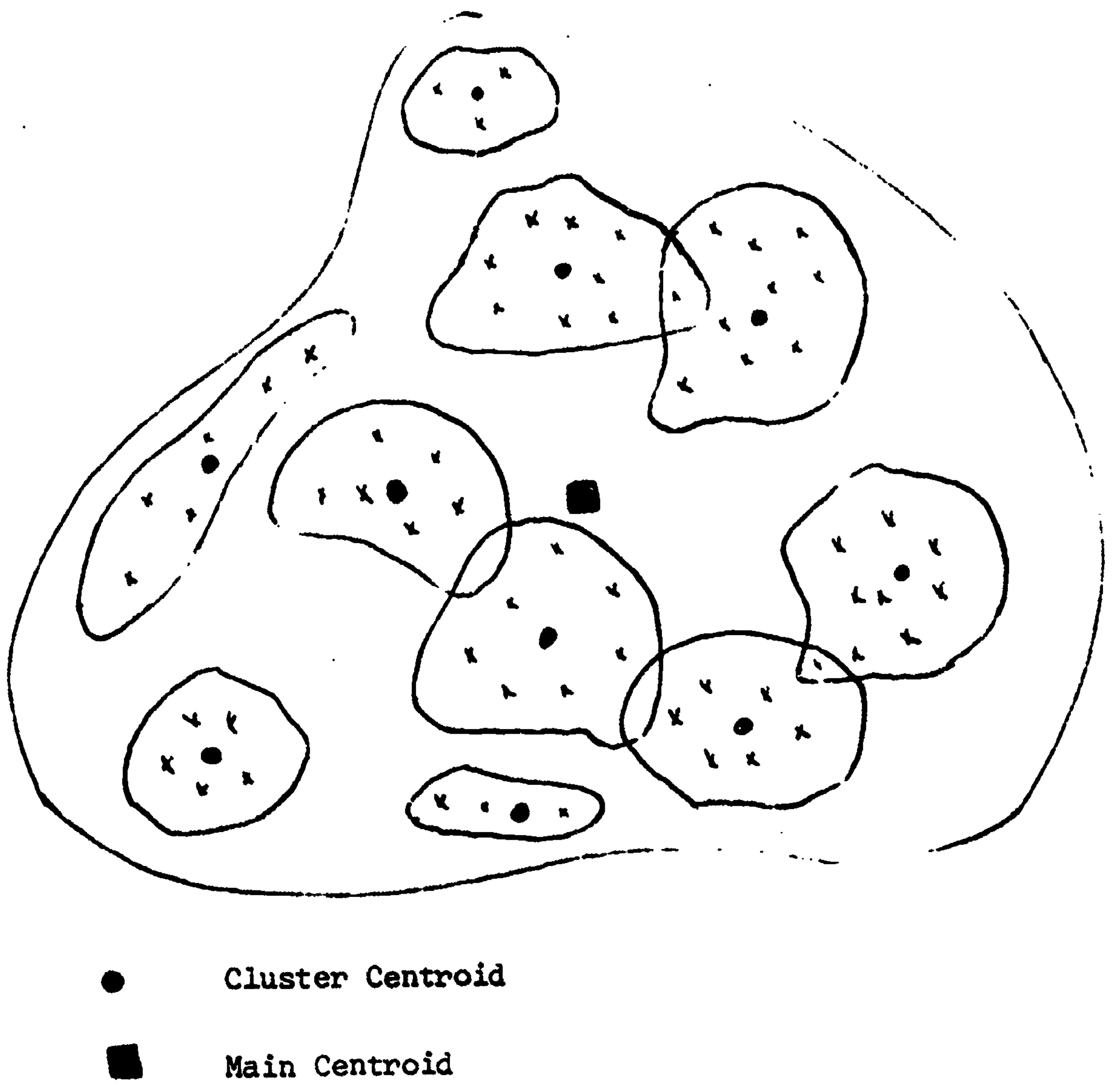
Two questions then arise: first, is it in fact the case that a separated document space leads to a good retrieval performance, and vice-versa that improved retrieval performance implies a wider separation of the documents in the space; second, is there a practical way of measuring the space separation. In practice, the expression of equation (1) is difficult to compute since the number of vector comparisons is proportional to n^2 for a collection of n documents.

For this reason, a clustered document space is best considered, where the documents are grouped into classes, each class being represented by a class centroid. A typical clustered document space is shown in Fig. 4, where the various document groups are represented by circles and the centroids by black dots located more or less at the center of the respective clusters.+ For a given document class K comprising m documents, each element of the centroid C may then be defined as the average weight of the same elements in the corresponding document vectors, that is

$$c_j = \frac{1}{m} \sum_{\substack{i=1 \\ D_i \varepsilon K}}^{m} d_{ij}. \qquad (2)$$

*In practice, the best performance is achieved by obtaining for each user a desired recall level (a specified proportion of the relevant items); at that recall level, one then wants to maximize precision by retrieving as few of the nonrelevant as possible.

+A number of well-known clustering methods exist for automatically generating a clustered collection from the term vectors representing the individual documents. [1]



Clustered Document Space

Fig. 4

Corresponding to the centroid of each individual document cluster, a centroid may be defined for the whole document space. This main centroid, represented by a small rectangle in the center of Fig. 4, may then be obtained from the individual cluster centroids in the same manner as the cluster centroids are computed from the inuividual documents. That is, the main centroid of the complete space is simply the average of the various cluster centroids.

In a clustered document space, the space density measure consisting of the sum of all pairwise document similarities, introduced earlier as equation (1), may be replaced by the sum of all similarity coefficients between each document and the main centroid, that is

$$Q = \sum_{i=1}^{n} s(C^*, D_i), \qquad (3)$$

where C^* denotes the main centroid. Whereas the computation of equation (1) requires n^2 operations, an evaluation of equation (3) is proportional to n.

Given a clustered document space such as the one shown in Fig. 4, it is necessary to decide what type of clustering represents most closely the separated space shown for the unclustered case in Fig. 3. If one assumes that documents that are closely related within a single cluster normally exhibit identical relevance characteristics with respect to most user queries, then the best retrieval performance should be obtainable with a clustered space exhibiting tight individual clusters, but large intercluster distances; that is,

a) the average similarity between pairs of documents within a single cluster should be maximized, while simultaneously

b) the average similarity between different cluster centroids is minimized.

The reverse obtains for cluster organizations not conducive to good performance where the individual clusters should be loosely defined, whereas the distance between different cluster centroids should be small.

In the remainder of this study, actual performance figures are given relating document space density to retrieval performance, and conclusions are reached regarding good models for automatic indexing.

2. Correlation between Indexing Performance and Space Density

The main techniques useful for the evaluation of automatic indexing methods are now well understood. In general, a simple straightforward process can be used as a base-line criterion — for example, the use of certain word stems extracted from documents or document abstracts, weighted in accordance with the frequency of occurrence (f_i^k) of each term k in document i. This method is known as term-frequency weighting. Recall-precision graphs can be used to compare the performance of this standard process against the output produced by more refined indexing methods. Typically, a recall-precision graph is a plot giving precision figures, averaged over a number of user queries, at ten fixed recall levels, ranging from 0.1 to 1.0 in steps of 0.1. The better indexing method will of course produce higher precision figures at equivalent recall levels.

One of the best automatic term weighting procedures evaluated as part of a recent study consisted of multiplying the standard term frequency weight f_i^k by a factor inversely related to the document frequency d_k of the term (the number of documents in the collection to which the term is assigned). [2] Specifically, if d_k is the document frequency of term k, the inverse document frequency IDF_k of term k may be defined as [3]:

$$(\mathrm{IDF})_k = \lceil \log_2 n \rceil - \lceil \log_2 d_k \rceil + 1.$$

A term weighting system proportional to ($f_i^k \cdot \mathrm{IDF}_k$) will assign the largest weight to those terms which arise with high frequency in individual documents, but are at the same time relatively rare in the collection as a whole.

It was found in the earlier study that the average improvement in recall and precision (average precision improvement at the ten fixed recall points) was about 14 percent for the system using inverse document frequencies over the standard term frequency weighting. The corresponding space density measurements are shown in Table 1 using two different cluster organizations for a collection of 424 documents in aerodynamics:

a) Cluster organization A is based on a large number of relatively small clusters, and a considerable amount of overlap between the clusters (each document appears in about two clusters on the average); the clusters are defined from the document-query relevance assessments, by placing into a common class all documents jointly declared relevant to a given user query.

b) Cluster organization B exhibits fewer classes (83 versus 155) of somewhat larger size (6.6 documents per class on the average versus 5.8 for cluster organization A); there is also much less overlap among the clusters (1.3 clusters per document versus 2.1). The classes are constructed by using a fast automatic tree-search algorithm due to Williamson. [4]

A number of space density measures are shown in Table 1 for the two cluster organizations, including the average similarity between the documents and the corresponding cluster centroids (factor x); the average similarity between the cluster centroids and the main centroid; and the average similarity between pairs of cluster centroids (factor y). Since a well-separated space

Type of Indexing	Cluster Organization A (155 clusters; 2.1 overlap)		Cluster Organization B (83 clusters; 1.3 overlap)	
	Standard Term Frequency Weights (f_i^k)	Term - Frequency with Inverse Doc. Freq. ($f_i^k \cdot IDF_k$)	Standard Term Frequency Weights (f_i^k)	Term - Frequency with Inverse Doc. Freq. ($f_i^k \cdot IDF_k$)
Recall-Precision Output (cf. Doc. December 73)	—	+14%	—	+14%
Average Similarity between Documents and Corresponding Cluster Centroids (x)	.712	.668 (-.044)	.650	.589 (-.061)
Average Similarity Between Cluster Centroids and Main Centroid	.500	.454 (-.046)	.537	.492 (-.045)
Average Similarity Between Pairs of Cluster Centroids (y)	.273	.209 (-.064)	.315	.252 (-.063)
Ratio y/x	$\frac{.273}{.712}$ = .383	$\frac{.209}{.668}$ = .318 (-19%)	$\frac{.315}{.650}$ = .485	$\frac{.252}{.589}$ = .428 (-12%)

Effect of Performance Improvement on Space Density

Table 1

corresponds to tight clusters (large x) and large differences between different clusters (small y), the ratio y/x can be used to measure the overall space density. [5]

It may be seen from Table 1, that all density measures are smaller for the indexing system based on inverse document frequencies; that is, the documents within individual clusters resemble each other less, and so do the complete clusters themselves. However, the "spreading out" of the clusters is greater than the spread of the documents inside each cluster. This accounts for the overall decrease in space density between the two indexing systems. The results of Table 1 would seem to support the notion that improved recall-precision performance is associated with decreased density in the document space.

The reverse proposition, that is, whether decreased performance implies increased space density may be tested by carrying out term weighting operations inverse to the ones previously used. Specifically, since a weighting system in inverse document frequency order produces a high recall-precision performance, a system which weights the terms directly in order of their document frequencies (terms occurring in a large number of documents receive the highest weights) should be correspondingly poor. In the output of Table 2, a term weighting system proportional to $(f_i^k \cdot DF_k)$ is used, where f_i^k is again the term frequency of term k in document i, and DF_k is defined as $10/(IDF)_k$. The recall-precision figures of Table 2 show that such a weighting system produces a decreased performance of about ten percent, compared with the standard.

The space density measurements included in Table 2 are the same as those in Table 1. For the indexing system of Table 2, a general "bunching up" of the space is noticeable, both inside the clusters and between clusters.

	Cluster Organization A (155 clusters; 2.1 overlap)		Cluster Organization B (83 clusters; 1.3 overlap)	
Type of Indexing	Standard Term Frequency Weights (f_i^k)	Term Frequency with Document Frequency ($f_i^k \cdot DF_k$)	Standard Term Frequency Weights (f_i^k)	Term Frequency with Document Frequency ($f_i^k \cdot DF_k$)
Recall- Precision Output	—	-10.1%	—	-10.1%
Average Similarity Between Documents and Corresponding Cluster Centroids (x)	.712	.741 (+.029)	.650	.696 (+046)
Average Similarity Between Cluster Centroids and Main Centroid	.500	.555 (+.055)	.537	.574 (+.037)
Average Similarity Between Pairs of Cluster Centroids (y)	.273	.329 (+.056)	.315	.362 (+.047)
Ratio y/x	$\frac{.273}{.712} = .383$	$\frac{.329}{.741} = .444$ (+16%)	$\frac{.315}{.650} = .485$	$\frac{.362}{.696} = .520$ (+7%)

Effect of Performance Deterioration on Space Density

Table 2

However, the similarity of the various cluster centroids increases more than that between documents inside the clusters. This accounts for the higher y/x factor by 16 and 7 percent for the two cluster organizations, respectively.

3. Correlation between Space Density and Indexing Performance

In the previous section it was shown that certain indexing methods which operate effectively in a retrieval environment are associated with a decreased density of the vectors in the document space, and contrariwise that poor retrieval performance corresponds to a space that is more compressed.

The relation between space configuration and retrieval performance may, however, also be considered from the opposite viewpoint. Instead of picking document analysis and indexing systems with known performance characteristics and testing their effect on the density of the document space, it is possible artifically to change the document space configurations in order to ascertain whether the expected changes in recall and precision are in fact produced.

The space density criteria previously given stated that a collection of small tightly clustered documents with wide separation between individual clusters should produce the best performance. The reverse is true of large nonhomogeneous clusters that are not well separated. To achieve improvements in performance, it would then seem to be sufficient to increase the similarity between document vectors located in the same cluster, while decreasing the similarity between different clusters or cluster centroids. The first effect is achieved by emphasizing the terms that are unique to only a few clusters, or terms whose cluster occurrence frequencies are highly skewed (that is, they occur with large occurrence frequencies in some clusters, and with much lower frequencies in many others). The second result is produced by deemphasizing terms that occur in many different clusters.

Two parameters may be introduced to be used in carrying out the required transformations [5]:

NC(k) The number of clusters in which term k occurs (a term occurs in a cluster if it is assigned to at least one document in that cluster);

and CF(k,j) the cluster frequency of term k in cluster j that is, the number of documents in cluster j in which term k occurs.

For a collection arranged into p clusters, the average cluster frequency $\overline{CF}(k)$ may then be defined from CF(k,j) as

$$CF(k) = \frac{1}{p} \sum_{j=1}^{p} CF(k,j).$$

Given the above parameters, the skewness of the occurrence frequencies of the terms may now be measured by a factor such as

$$F_1 = |\overline{CF}(k) - CF(k,j)|.$$

On the other hand, a factor F_2 inverse to NC(k) (for example, 1/NC(k)) can be used to reflect the rarity with which term k is assigned to the various clusters. By multiplying the weight of each term k in each cluster j by a factor proportional to $F_1 \cdot F_2$ a suitable spreading out should be obtained in the document space. Contrariwise, the space will be compressed when a multiplicative factor proportional to $1/F_1 \cdot F_2$ is used.

The output of Table 3 shows that a modification of term weights by the $F_1 \cdot F_2$ factor produces precisely the anticipated effect: the similarity between documents included in the same cluster (factor x) is now greater, whereas the similarity between different cluster centrcids (factor y) has

	Cluster Organization A (155 clusters; 2.1 overlap)		Cluster Organization B (83 clusters; 1.3 overlap)	
	Standard Cluster Density (term frequency weights)	High Cluster Density (emphasis of low frequency and skewed terms)	Standard Cluster Density (term frequency weights)	High Cluster Density (emphasis of low frequency and skewed terms)
Average Similarity between Documents and their Centroids (x)	.712	.730 (+.018)	.650	.653 (+.003)
Average Similarity between Cluster Centroids and Main Centroid	.500	.477 (-.023)	.537	.528 (-.009)
Average Similarity between Pairs of Centroids (y)	.273	.229 (-.044)	.315	.281 (-.034)
Ratio y/x	$\frac{.273}{.712} = .383$	$\frac{.229}{.730} = .314$ (-18%)	$\frac{.315}{.650} = .485$	$\frac{.281}{.653} = .430$ (-11%)
Recall-Precision Comparison	——	+2.6%	——	+2.3%

Effect of Low Cluster Density on Performance

Table 3

decreased. Overall, the space density measure (y/x) decreases by 18 and 11 percent respectively for the two cluster organizations. The average retrieval performance for the spread-out space shown at the bottom of Table 3 is improved by a few percentage points.

The corresponding results for the compression of the space using a transformation factor of $1/F_1 \cdot F_2$ are shown in Table 4. Here the similarity between documents inside a cluster decreases, whereas the similarity between cluster centroids increases. The overall space density measure (y/x) increases by 11 and 16 percent for the two cluster organizations compared with the space representing the standard term frequency weighting. This dense document space produces losses in recall and precision performance of 12 to 13 percent.

Taken together, the results of Tables 1 to 4 indicate that retrieval performance and document space density appear inversely related, in the sense that effective (questionable) indexing methods in terms of recall and precision are associated with separated (compressed) document spaces; on the other hand, artificially generated alterations in the space densities appear to produce the anticipated changes in performance.

The foregoing evidence thus confirms the usefulness of the "term discrimination" model and of the automatic indexing theory based on it. These questions are examined briefly in the remainder of this study.

4. The Discrimination Value Model

For some years, a document indexing model known as the term discrimination model has been used experimentally. [2,6] This model bases the value of an index term on its "discrimination value" DV, that is, on an index which measures the extent to which a given term is able to increase the differences among document vectors when assigned as an index

	Cluster Organization A (155 clusters; 2.1 overlap)		Cluster Organization B (83 clusters; 1.3 overlap)	
	Standard Cluster Density (term frequency weights)	High Cluster Density (emphasis on high frequency and even terms)	Standard Cluster Density (term frequency weights)	High Cluster Density (emphasis on high frequency and even terms)
Average Similarity between Documents and their Centroids (x)	.712	.681 (-.031)	.650	.645 (-.005)
Average Similarity between Cluster Centroids and Main Centroid	.500	.523 (+.023)	.537	.531 (-.006)
Average Similarity between Pairs of Centroids (y)	.273	.290 (+.017)	.315	.364 (+.049)
Ratio y/x	$\frac{.273}{.712} = .383$	$\frac{.290}{.681} = .426$ (+11%)	$\frac{.315}{.650} = .485$	$\frac{.364}{.645} = .564$ (+16%)
Recall-Precision Comparison	—	-12.4%	—	-13.3%

Effect of High Cluster Density on Performance

Table 4

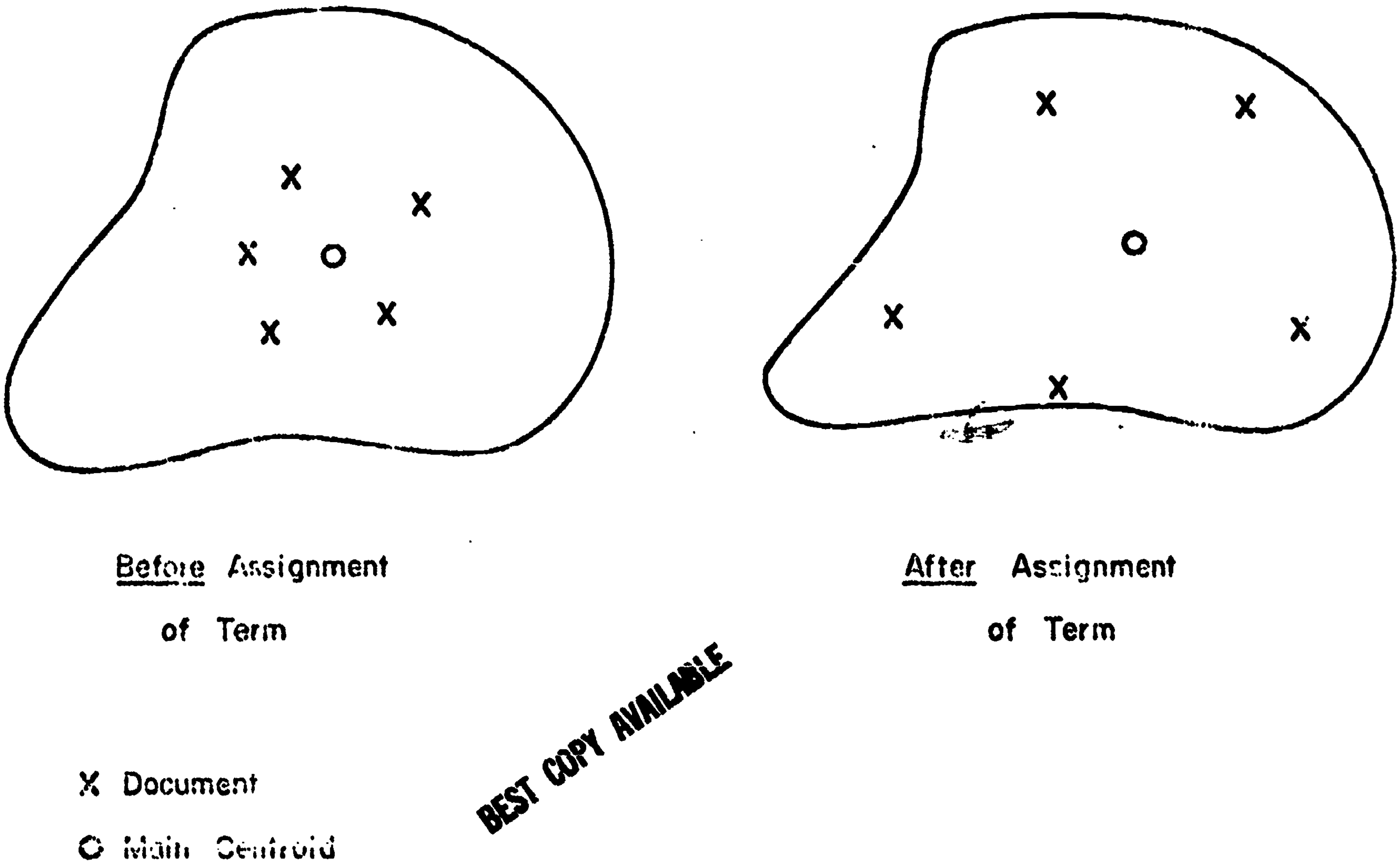
term to a given collection of documents. A "good" index term — one with a high discrimination value — decreases the similarity between documents when assigned to the collection, as shown in the example of Fig. 5. The reverse obtains for the "bad" index term with a low discrimination value.

To measure the discrimination value of a term, it is sufficient to take the difference in the space densities before and after assignment of the particular term. Specifically, let the density of the complete space be measured by a function Q such as that of equation (3); that is, by the sum of the similarities between all documents and the space centroid. The contribution of a given term k to the space density may be ascertained by computing the function

$$DV_k = Q_k - Q, \tag{4}$$

where Q_k is the compactness of the document space with term k deleted from all document vectors. If term k is a good discriminator, valuable for content identification, then $Q_k > Q$, that is, the document space after removal of term k will be more compact (because upon assignment of that term to the documents of a collection the documents will resemble each other less and the space spreads out). Thus for good discriminators $Q_k - Q > 0$; the reverse obtains for poor discriminators for which $Q_k - Q < 0$.

Because of the manner in which the discrimination values are defined, it is clear that the good discriminators must be those with uneven occurrence frequency distributions which cause the space to spread out when assigned by decreasing the similarity between the individual documents. The reverse is true for the bad discriminators. A typical list including the ten best terms and the ten worst terms in discrimination value order (in order by the



Before Assignment of Term

After Assignment of Term

Operation of

Good Discriminating Term

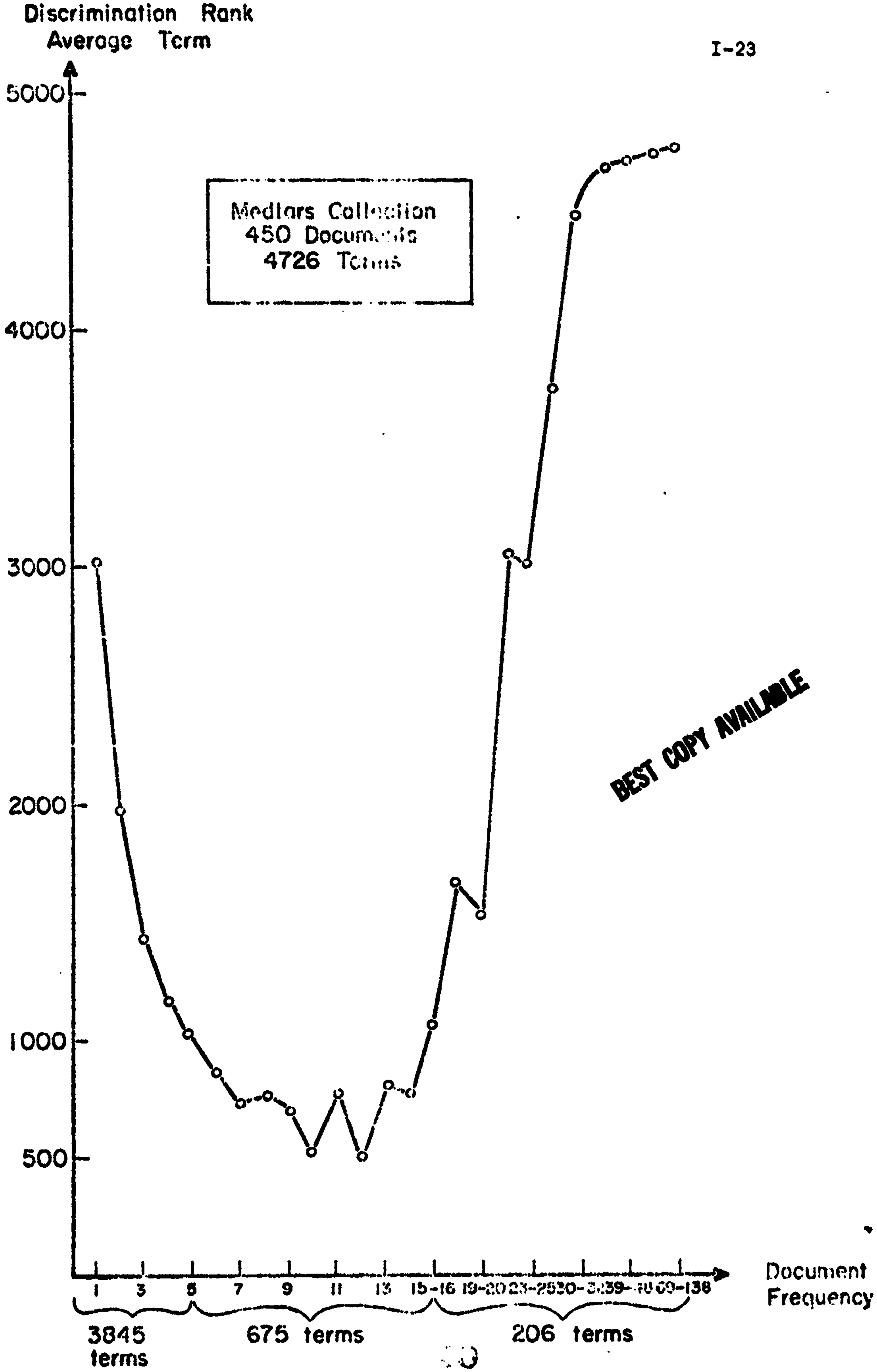
Fig. 5

Q_k - Q value) is shown in Table 5 for a collection of 425 articles in world affairs from Time magazine. A total of 7569 terms are used for this collection, exclusive of the common English function words that have been deleted.

In order to translate the discrimination value model into a possible theory of indexing, it is necessary to examine the properties of good and bad discriminators in greater detail. Fig. 6 is a graph of the terms assigned to a sample collection of 450 documents in medicine, presented in order by their document frequencies. For each class of terms — those of document frequency 1, document frequency 2, etc. ... — the average rank of the corresponding terms is given in discrimination value order (rank 1 is assigned to the best discriminator and rank 4726 to the worst term for the 4726 terms of the medical collection).

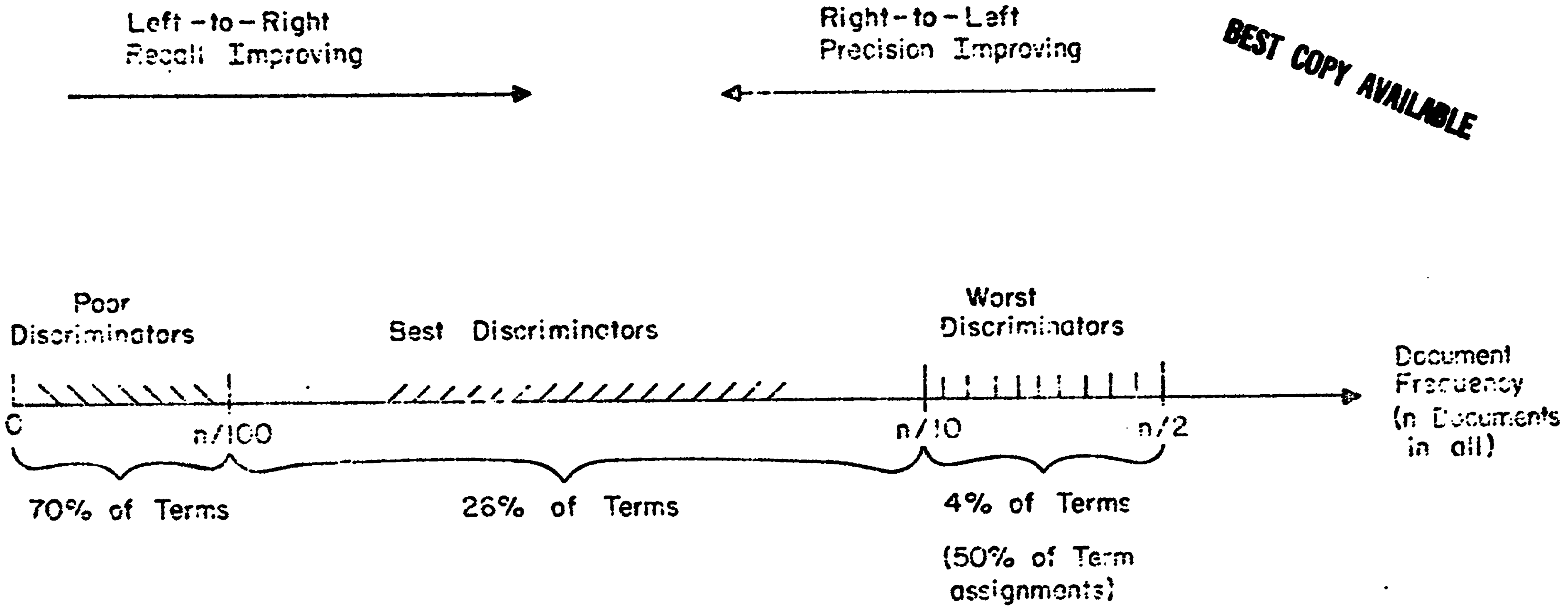
Fig. 6 shows that terms of low document frequency — those that occur in only one, or two, or three documents — have rather poor average discrimination ranks. The several thousand terms of document frequency 1 have an average rank exceeding 3000 out of 4726 in discrimination value order. The terms with very high document frequency — at least one term in the medical collection occurs in as many as 138 documents out of 450 — are even worse discriminators; the terms with document frequency greater than 25 have average discrimination values in excess of 4000 in the medical collection. The best discriminators are those whose document frequency is neither too low nor too high.

The situation relating document frequency to term discrimination value is summarized in Fig. 7. The 4 percent of the terms with the highest document frequency, representing about 50 percent of the total term assignments to the documents of a collection, are the worst discriminators. The 7) percent of



Average Discrimination Value Rank of Terms

Fig. 6



Summarization of Discrimination Value of

Terms in Frequency Ranges

Fig. 7

Good Terms		Poor Terms	
1.	Buddhist	7560.	Work
2.	Diem	7561.	Lead
3.	Lao	7562.	Red
4.	Arab	7563.	Minister
5.	Viet	7564.	Nation
6.	Kurd	7565.	Party
7.	Wilson	7566.	Commune
8.	Baath	7567.	U.S.
9.	Park	7568.	Govern
10.	Nenni	7569.	New

Terms in Discrimination Value Order

(1963 Time Magazine)

Table 5

the terms with the lowest document frequency are generally poor discriminators. The best discriminators are the 25 percent whose document frequency lies approximately between n/100 and n/10 for n documents.

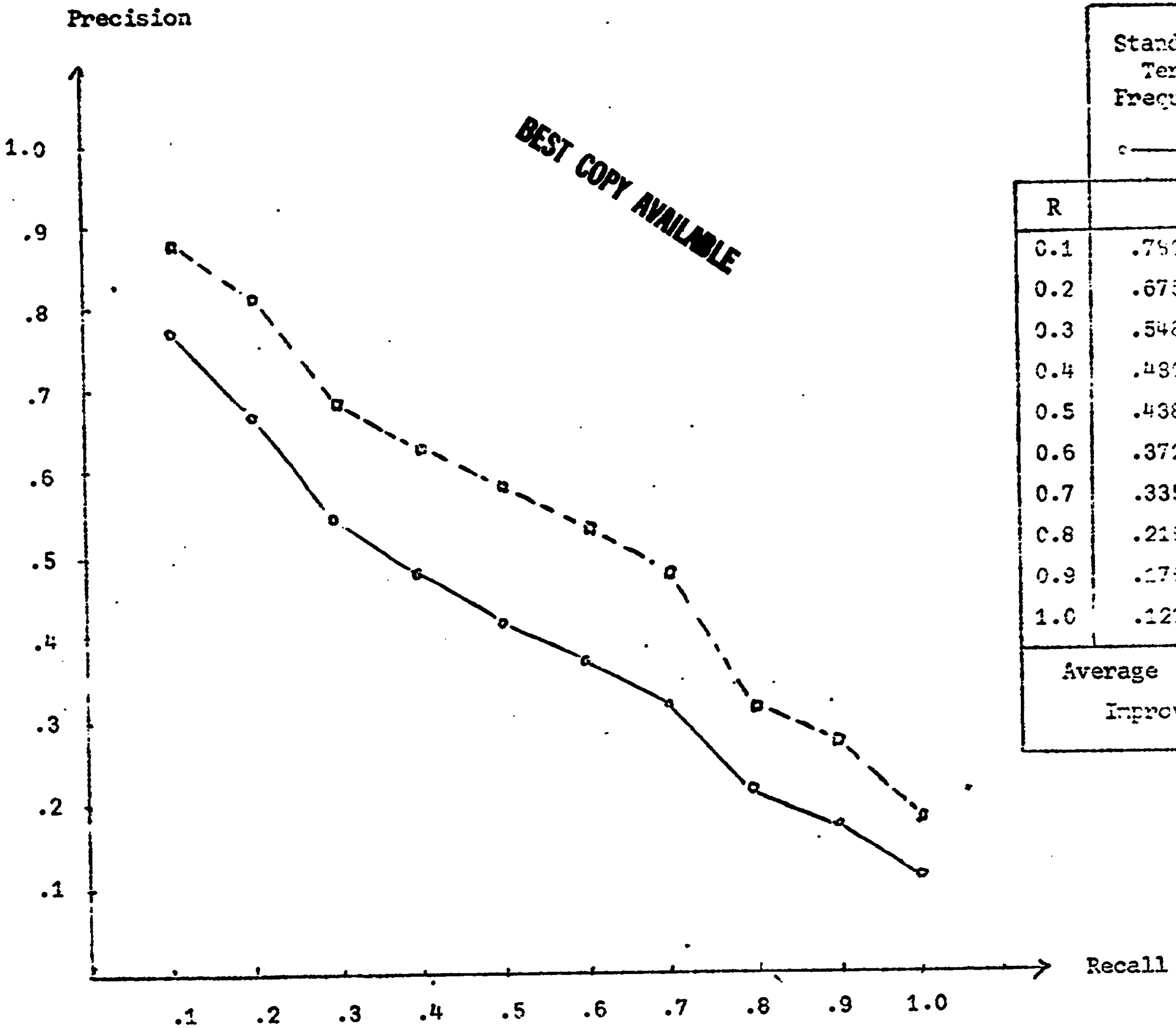
If the model of Fig. 7 is a correct representation of the situation relating to term importance, the following indexing strategy results [6,7]:

a) Terms with medium document frequency should be used for content identification directly, without further transformation.

b) Terms with very high document frequency should be moved to the left on the document frequency spectrum by transforming them into entities of lower frequency; the best way of doing this is by taking high-frequency terms and using them as components of indexing phrases — a phrase such as "programming language" will necessarily exhibit lower document frequency than either "program", or "language" alone.

c) Terms with very low document frequency should be moved to the right on the document frequency spectrum by being transformed into entities of higher frequency; one way of doing this is by collecting several low frequency terms that appear semantically similar and including them in a common term (thesaurus) class. Each thesaurus class necessarily exhibits a higher document frequency than any of the component members that it replaces.

The indexing theory which consists in using certain elements extracted from document texts directly as index terms, combined with phrases made up of high frequency components and thesaurus classes defined from low frequency elements has been tested using document collections in aerodynamics (CRAN), medicine (MED), and world affairs (TIME). [2,6,7] A typical recall-precision plot showing the effect of the right-to-left phrase transformation is shown in Fig. 8 for the Medlars collection of 450 medical documents. When recall is plotted against precision, the curve closest to the upper right-hand corner



	Standard Term Frequency ○———○	Phrase Assignment □———□
R	Precision	
0.1	.7691	.8811
0.2	.6750	.8149
0.3	.5481	.6992
0.4	.4807	.6481
0.5	.4384	.5930
0.6	.3721	.5450
0.7	.3357	.4867
0.8	.2195	.3263
0.9	.1766	.2767
1.0	.1230	.1989
Average Improvement		+ 39%

Average Recall-Precision Comparison for Phrases

(Medlars : 450 Documents, 24 Queries)

Fig. 8

of the graph (where both recall and precision are close to 1) reflects the best performance. It may be seen from Fig. 8 that the replacement of the high-frequency nondiscriminators by lower frequency phrases improves the retrieval performance by an average of 39 percent (the precision values at the ten fixed recall points are greater by an average of 39 percent).

The performance of the right-to-left (phrase) transformation and left-to-right (thesaurus) transformation is summarized in Table 6 for the three previously mentioned test collections. The precision values obtainable are near 90 percent for low recall, between 40 and 70 percent for medium recall, and between 15 and 45 percent at the low recall end of the performance spectrum. The overall improvement obtainable by phrase and thesaurus class assignments over the standard term frequency process using only the unmodified, single terms ranges from 17 percent for the world affairs collection to 50 percent for the medical collection.

A conclusive proof relating the space density analysis and the resulting document frequency indexing model to optimality in the retrieval performance cannot be furnished. However, the model appears to perform well for collections in several different subject areas, and the performance results produced by applying the theory have not in the authors' experience been surpassed by any other manual or automatic indexing and analysis procedures tried in earlier experiments. The model may then lead to the best performance obtainable with ordinary document collections operating in actual user environments.

BEST COPY AVAILABLE

CRAN 424	MED 450	TIME 425
Automatic Phrases vs. Standard Term Frequency +32%	Automatic Phrases vs. Standard Term Frequency +39%	Automatic Phrases vs. Standard Term Frequency
Automatic Phrases Plus Thesaurus vs. Standard Run +33%	Automatic Phrases Plus Thesaurus vs. Standard Run +50%	Automatic Phrases Plus Thesaurus vs. Standard Run
Best Precision Low Recall 0.89 Medium Recall 0.43 High Recall 0.13	Best Precision Low Recall 0.88 Medium Recall 0.61 High Recall 0.23	Best Precision Low Recall 0.85 Medium Recall 0.70 High Recall 0.45

Summary of Recall-Precision Evaluation (Three Collections)

Table 6

References

[1] G. Salton, Automatic Information Organization and Retrieval, McGraw Hill Book Co., New York, 1968, chapter 4.

[2] G. Salton and C.S. Yang, On the Specification of Term Values in Automatic Indexing, Journal of Documentation, Vol. 29, No. 4, December 1973, p. 351-372.

[3] K. Sparck Jones, A Statistical Interpretation of Term Specificity and its Application to Retrieval, Journal of Documentation, Vol. 28, No. 1, March 1972, p. 11-20.

[4] R.E. Williamson, Real-time Document Retrieval, Cornell University Ph.D. Thesis, Department of Computer Science, June 1974.

[5] A. Wong, An Investigation of the Effects of Different Indexing Methods on the Document Space Configuration, Scientific Report No. ISR-22, Department of Computer Science, Cornell University, to appear.

[6] G. Salton, A Theory of Indexing, Technical Report No. TR-203, Department of Computer Science, Cornell University, March 1974.

[7] G. Salton, C.S. Yang, and C.T. Yu, Contribution to the Theory of Indexing, Proc. IFIP Congress-74, Stockholm, August 1974.

An Investigation on the Effects cf Different Indexing Methods on the Document Space Configuration

Anita Wong

Abstract

An attempt is made on the present study to gain a better understanding of the document space configuration through the use of clustered document collections and different indexing methods.

1. Introduction

Previous work in automatic indexing and clustering in information retrieval has mostly been done with the thought of improving the recall and/or precision of the search result. Not too much work has been done to gain a fuller understanding of the document space configuration itself, presumably because this is not directly related to the improvement of the effectiveness of the system. However it is quite likely that the configuration of the document space does correlate in some way with the effectiveness of the system.

It is natural for documents that are related to be closely similar to each other.* But is this really the case for any indexing method? Or is it possible that documents that are related are scattered throughout the document space and surrounded by extraneous documents which are more or less closely packed in groups?

* Experimental results were performed by Jones [1].

The problem would be easier to answer if the meaning of closeness were better defined. Closeness should bear a different meaning in different systems, depending on the way the documents are retrieved. On a book shelf, two books are said to be close together if they are physically close together. Close is defined in this way because whenever one book is located, the other is also found. In automatic retrieval, closeness would be proportional to the matching function used for document retrieval. The physical distance between the documents is less important in this respect.

In the SMART system, a document is retrieved by a query if the similarity between the document and query is high. If the similarity relation is assumed transitive, then documents relevant to the same query should be similar to each other. It is therefore inconceivable that any indexing method should place the related documents in any way other then "close" together. However merely placing the related documents close together does not necessarily guarantee good system performance. The unrelated documents should be farther apart then the related ones. In other words, ideally, documents should form clusters which do not overlap; documents within a cluster are related while those in different clusters are not; and the distance between two documents within a cluster is shorter than two documents belonging to two different clusters. Consequently, a good indexing method should index the documents in such a way that the related documents are close together and non-related ones further apart.

Many different indexing methods were tested over the years in the SMART system, and recall and precision figures were generated. It is not known if the indexing methods that produce good performance do in fact place the documents in the way predicted above, or if the changes in configuration of the document space can be explained in some other way. It is the aim of the present work to elucidate the relationship between the configuration of the document space with the performance of the system with different indexing methods.

2. Methodology

An obvious way to solve the problem is to look at each document, and at its relation with the other documents. This requires a computation of the correlation of every document with every other documents, thus obtaining for each indexing method a full document-document matrix. These matrices are then compared row-wise in order to ascertain the relative changes of the documents with respect to each other. This method is not employed here because the number of documents involved is usually large. Instead, the documents are grouped, and each group is treated as an entity with respect to the documents not in the same group. Judging the movements of groups of documents instead of individual items alone may be justified because it is rather pointless to look at the motion of each document with respect to every other, since there are so many documents that individual effects would be hardly noticeable.

The discrimination value model [2] has been found to give improvements in search performance. However, the discrimination values are determined by lowering the sum of the correlations between the documents and their centroid; in other words, the discrimination value is a function of the distance between document vectors. The application of the discrimination values would tend to increase the average distance between documents, regardless of the relations between the documents. But it may be unreasonable to have related documents farther apart. Consequently, it is important to relate the average increase in distance between similar documents to the average increase in distance between all documents.

Since the investigation is performed on clustered document collections, the clustering methods to be used are discussed first.

Previous experiments were done with clustering methods that produce clusters of different sizes with a large variance.* It was found that a large cluster is represented by a longer centroid vector. In some cases, the centroid vectors were so long that they each included 75% of the terms occurring in the collection. The correlations between these centroids, being high, have become less meaningful in distinguishing one centroid from another. The problem could be overcome by using a different method in forming the centroid vectors as in Murray and Kerchner [3], [4], or by deleting some common terms. But the centroids would then lack some of the terms occurring in the documents and thus the centroid would not be the true centers of the document clusters. For this reason clustering algorithms that produce smaller clusters are considered.

The clustering methods to be used are:

A. For each query, one cluster is constructed. The cluster contains documents that are relevant to that query. This method is chosen because the clusters are easy to obtain in an experimental environment and also because it produces small clusters as shown in Table 1. This method will be referred to as method RCL.

* The centroid vector of a cluster is formed by summing the normalized document vectors. Let D_i $i = 1 \ldots m$ be documents belonging to cluster c then the centroid vector for

$$c = \sum_{i=1}^{m} \frac{D_i}{|D_i|} .$$

B. Williamson's Clustering Algorithm. This method is essentially a tree building procedure, which has a bound on the number of sons each node may have. This method is used because it also produces small clusters. This method will be referred to as method SKIP. [5]

Clustering Method	RCL	SKIP
Number of Clusters	155	83
Average number of documents in one cluster	5.8	6.6
Number of documents in the largest cluster	22	9
Number of documents in the smallest cluster	3	4
Sum of the number of documents in all clusters	900	547

Statistics for the two Clustering Methods

Table 1

3. Cluster Measurements

A number of measurements are performed on the clusters.

Notation:

c = the main centroid of the entire collection

c_i = the ith cluster centroid

R_j = the jth document

D_j = the jth cluster

$|D_j|$ = the number of documents in D_i

N = the number of clusters.

The measurements are:

1. The average correlation of the documents in cluster D_i with their centroid, C_i

$$A_i = \frac{\sum_{R_j + D_i} \text{cosine}(R_j, C_i)}{|D_j|}$$

The average for all the clusters

$$a = \sum_i A_i / N$$

2. The correlation between cluster centroid C_i with the main centroid, C

$$B_i = \text{cosine}(C_i, C)$$

and the average of the B_i's

$$b = \Sigma B_i / N$$

3. The correlation between two cluster centroids

$$C_{ij} = \text{cosine}(C_i, C_j)$$

and the average of the C_{ij}'s

$$C = \sum_i \sum_j C_{ij} / N^2$$

4. The ratio: $\dfrac{\text{the ave. corr. of the docs, with their centroid}}{\text{the ave. corr. between cluster centroids and main centroid}}$

$$Q_1 = a/b$$

5. The ratio: $\dfrac{\text{Ave. corr. of the documents with their centroid}}{\text{Ave. corr. between cluster centroids}}$

$$Q_2 = a/c.$$

4. The Experiment

The experiment was performed using the Cranfield 424 Thesaurus Collection. The collection was first clustered using Williamson's Algorithm SKIP, and then using the query relevant method, RCL. The Q values for these two clustered document collections were calculated for the different indexing methods listed below.

1. Cranfield 424 Thesaurus Collection.
2. Cranfield 424 Thesaurus Collection with the application of discrimination values. The concept weight of the document vectors were multiplied by the discrimination values rescaled to an effective range.
3. Cranfield 424 Thesaurus Collection with the application of inverse document frequency. The concept-weight of the document vectors were modified by multiplying a function inversely proportional to the document frequency (DF) of the term, i.e. new concept weight = old concept weight x $\left[\log \frac{N}{DF} + 1\right]$. This model emphasizes the low frequency terms, and deemphasizes the high frequency terms.
4. An indexing method that does not perform as well as the control method (method 1). To create the collection, the Cranfield 424 Thesaurus collection is modified by deleting one hundred terms which have discrimination values higher than 0.04. By deleting the high discrimination value terms, it is evident that the document vectors will be moved towards each other. However the essential changes of the orientation of the document vectors have yet to be determined. This method is referred to as HDVD.
5. An indexing method created for the purpose of this work. It is believed that a good indexing method would place the documents into natural clusters with the inter-cluster distance relatively large. To achieve this result, the documents were modified so as to diminish the distance between documents within each cluster,

while increasing the length of the inter-cluster distances. To diminish the distance between documents within a cluster, the terms to be emphasized must be those that are unique to a few clusters, that is, terms that have low cluster frequency should be emphasized to increase the correlation between documents within those few clusters. To decrease the correlation between clusters, terms that occur in relatively more clusters are deemphasized. Using these two criteria, a value, val(t) is determined for each concept t. The document collection is modified by multiplying the original concept weights by this value. The actual procedure is as follows:

1) For each cluster j, the document frequency of each term t in the cluster is found. This is denoted by CLUSFREQ(t,j) for term t and cluster j.

2) For each term the number of different clusters in which it occurs (i.e. the cluster frequency of each term) is found. It is denoted by NCLUS(t).

3) The value, val(t), for term t is determined by the equation: val(t) = TMULT(t) x DAC(t), where TMULT(t) is a step function which is inversely proportional to the cluster frequency of term t and DAC(t) is a function proportional to the skewness of a term with respect to the clusters. They are defined by the following equations:

Given that STEP and LOWLIM are some integers

$$TMULT\ (t) = \begin{array}{llrcl} 2 & \text{if} & 1 \le & NCLUS(t) & \le STEP \\ 1.75 & \text{if} & STEP < & NCLUS(t) & \le 2 \times STEP \\ 1.50 & \text{if} & 2 \times STEP < & NCLUS(t) & \le 3 \times STEP \\ 1.25 & \text{if} & 3 \times STEP < & NCLUS(t) & \le 4 \times STEP \\ 1.00 & \text{if} & 4 \times STEP < & NCLUS(t) & \le LOWLIM \\ .50 & \text{if} & LOWLIM < & NCLUS(t) & \end{array}$$

$$\text{AVECLUS}(t) = \Big(\sum_{j \in \text{cluster}} \text{CLUSFREQ}(t,j)\Big)/\text{NCLUS}(t).$$

$$\text{DAC}(t) = \Big(\sum_{j \in \text{cluster}} \Big[\, |\text{AVECLUS}(t) - \text{CLUSFREQ}(t,j)| + 1 \Big] \Big\}/\text{NCLUS}(t)$$

AVECLUS(t) is the average document frequency for term t and DAC(t) is the average of the sum of the deviation of document frequency in the cluster from the average.

Examples:

1. a term t occurs in 3 clusters once in each:

   CLUSFREQ(t,i) = 1   i = 1,2,3

   AVECLUS(t) = 3/3 = 1

   DAC(t) = (1 + 1 + 1)/3 = 1

2. a term occurs in 3 clusters.

   CLUSFREQ(2,1) = 3

   CLUSFREQ(t,2) = 1

   CLUSFREQ(t,3) = 1

   AVECLUS(t) = (3 + 1 + 1)/3 = 1.2.

   $$\begin{aligned} \text{DAC}(t) &= (|1.7 - 3| + 1) + (|1.7 - 1| + 1) + \\ &\quad (|1.7 - 1| + 1)\}/3 \\ &= (2.3 + 1.7 + 1.7)/3 = 1.9. \end{aligned}$$

The collection thus modified will be referred to as MOD (STEP,LOWLIM).

The values 5 and 50 were used for STEP and LOWLIM. They were chosen to make TMULT > 1 for approximately half of the terms in the collection clustered using "SKIP". With the apparent better result obtained from the Relevant clustered collection, STEP and LOWLIM were changed to 3 and 38 respectively, for the cluster collection using SKIP, so that the number of terms with the same TMULT value is approximately the same for SKIP MOD(3,38) and RCL MOD(5,50).

6. For the sake of completeness another indexing method inverse of the MOD(5,50) was tried to move the documents within a cluster further away from each other as well as the clusters to each other.

1. MODI(1)

Similar to the MOD method the mod inverse collection is obtained by multiplying the original concept weights of the document vectors by a step function IVAL(t), defined as:

$$\text{IVAL(t)} = \text{ITMULT(t)}.$$

$$\text{ITMULT(t)} = \begin{cases} .5 & \text{if} \quad 1 \leq \text{NCLUS(t)} \leq 20 \\ 1.0 & \text{if} \quad 20 < \text{NCLUS(t)} \leq 50 \\ 2.0 & \text{if} \quad 50 < \text{NCLUS(t)} \end{cases}$$

2. MODI(2)

In MODI(2) the skewness factor, DAC(t), in MOD is also taken into consideration. IVAL(t) is defined as:

$$\text{IVAL(t)} = \text{ITMULT(t)} \times (4/\text{DAC(t)})$$

DAC(t) ranges from 1 to 4, thus 4 is picked here to keep (4/DAC(t)) in the same range.

5. The Results

The results can be summarized by Tables 2, 3 which is broken down into Tables 4 to 8 for clarity. The quantities a, b, c used here are the same as those in Tables 2 and 3 and explained in page 6.

	control	Dis	IDF	MOD(5, 50)	MOD(3, 38)	HDVD	MODI(1)	MODI(2)
a	.65	.603	.589	.649	.653	.664	.690	.645
b	.537	.484	.492	.522	.528	.611	.599	.581
c	.315	.237	.252	.274	.281	.375	.385	.364
Q_1	1.21	1.25	1.20	1.24	1.24	1.06	1.15	1.11
Q_2	2.0	2.5	2.3	2.3	2.32	1.77	1.79	1.77

The Summary of the Results for SKIP Clustered Collection

Table 2

	control	Dis	IDF	MOD(5, 50)	HDVD	MODI(1)	MODI(2)
a	.712	.689	.668	.73	.712	.725	.681
b	.50	.433	.454	.477	.579	.577	.523
c	.273	.192	.209	.229	.336	.312	.290
Q_1	1.42	1.6	1.57	1.53	1.23	1.3	1.1
Q_2	2.6	3.5	3.2	3.1	2.1	2.32	2.35

The Summary of the Results for RCL Clustered Collection

Table 3

	SKIP			RCL		
	a	b	c	a	b	c
control	.65	.537	.315	.712	.50	.273
dis	.603	.484	.237	.689	.433	.192
control / dis	1.08	1.11	1.33	1.03	1.15	1.42

Control vs Discrimination Values

Table 4

1. Discrimination Value Model

With the application of the discrimination values the average inter-cluster correlation (quantity b) and the average cluster and centroid correlation (quantity c) were lower than the corresponding ones for the control. This implies that the distances between clusters are lengthened with the application of the discrimination values, this results in a more spread out space. However, the document and cluster centroid correlations (quantity a) are also smaller, that is, within a cluster the discrimination values also cause the documents to spread out. Nevertheless, the control/discrimination value ratio for quantity a is smaller than those for quantity b and c in Table 4, implying that the expansion within a cluster is comparatively less than the expansion of the space itself. Furthermore, the Q_1 and Q_2 values of Table 2 and 3 for the discrimination value model are larger than the corresponding ones for the control. Therefore, with the use of discrimination values the space is more spread out and although the absolute sizes of the clusters are larger, relative to the size of the entire space the clusters are smaller.

2. Inverse Document Frequency

With the application of inverse document frequency all three quantities a, b and c are smaller, again indicating a more spread out document space. Except for the Q_1 value for SKIP's IDF collection the Q values for the IDF are larger than the control. The exception may be due to the non-relevant overlapping of SKIP's clustered collection. Aside from this exception, the results suggest the same conclusion as the results for the discrimination value collection.

	SKIP			RCL		
	a	b	c	a	b	c
control	.650	.537	.315	.712	.50	.273
IDF	.589	.492	.252	.668	.454	.209
$\frac{\text{control}}{\text{IDF}}$	1.10	1.09	1.21	1.07	1.10	1.31

Control vs Inverse Document Frequency
Table 5

3. The HDVD Collection

A surprising result in this case is that for relevant cluster quantity a, the average correlation between documents and their cluster is the same as that in the control collection. Although the actual correlations do vary for each cluster, on the average the clusters do not expand nor contract. In spite of the fact that with SKIP's clustered collection quantity as for the HDVD method is the largest, it is only a moderate increase over the control collection. It seems to denote that the discriminating term affects the inter-cluster distances more than the clusters themselves.

With the HDVD method all three quantities are larger or stay unchanged. The control/HDVD ratio is largest for quantity a, showing that the least changes occur in the individual clusters. Both Q values are decreased. All the changes were opposite from what was observed in the previous two cases. With the entire document space contracted and the individual clusters unchanged or less contracted. The HDVD method results in forming larger clusters with respect to the document space than the control collection.

	SKIP			RCL		
	a	b	c	a	b	c
control	.650	.537	.315	.712	.50	.273
HDVD	.664	.611	.375	.712	.579	.336
$\frac{\text{control}}{\text{HDVD}}$	.91	.88	.84	1.0	.86	.81

Control and HDVD

Table 6

	SKIP MOD (5, 50)			SKIP MOD (3, 38)			RCL MOD (5, 50)		
	a	b	c	a	b	c	a	b	c
control	.650	.537	.315	.650	.537	.315	.712	.50	.273
MOD	.649	.522	.274	.653	.528	.281	.73	.477	.229
$\frac{\text{control}}{\text{MOD}}$	1.0	1.03	1.5	1.0	1.02	1.12	.98	1.05	1.20

Control and the MOD Methods

Table 7

4. The MOD Methods

This method was created to decrease the distances between documents within a cluster and to increase the distances between clusters. Even though the control/MOD ratio of quantity a for all three cases, is less than or equal to one, only in one case-relevant cluster MOD (5,50) - is the size of the clusters significantly decreased. Of the other two cases, one is decreased slightly, and the other is increased slightly. However, quantities b and c were decreased so that the inter-cluster distances of all three cases increased significantly enough to produce larger Q_1 and Q_2 values.

	SKIP			RCL		
	a	b	c	a	b	c
control	.650	.537	.315	.712	.50	.273
MODI (1)	.690	.599	.385	.725	.557	.312
MODI (2)	.645	.581	.364	.681	.523	.290
$\frac{\text{control}}{\text{MODI (1)}}$	.94	.89	.82	.98	.90	.88
$\frac{\text{control}}{\text{MODI (2)}}$	1.0	.92	.87	1.1	.96	.94

Control vs MODI's

Table 8

5. The MODI Methods

From the control/MODI ratios we can see that the inter-cluster distances (quantities b and c) are lengthened. The results for MODI(2) is more satisfying since the cluster sizes increased as we hoped. Although the cluster sizes for MODI(2) decreased slightly, the MODI methods do give a smaller Q_1 and Q_2 values as we expected.

6. Conclusions

1. The fact that the distance between two centroids increases implies that on the average the distance between any document in one centroid and any other document in another centroid increases also. With the application of either the discrimination values or the inverse document

frequency, the distance between documents increased. Moreover, the MOD collections were constructed such that the inter-cluster distances are larger. All these indexing methods are found to have better search performance than the control method, as shown in Table 9. On the other hand, the HDVD model and the MODI models show that a more contracted document space produces deterioration in search performance. For the discrimination model, the result that the distances between document increase is obvious. However, it is not obvious for the inverse document frequency model, where the correlations between documents are not taken into consideration during construction. It can be concluded that a spread out document space is beneficial for the retrieval performance.

2. Indexing methods that have been found to produce improvements in search performance as measured by recall and precision, do have relatively more compact clusters with respect to the entire collection space. This conclusion is justified by the observation that, in all cases (the discrimination value, inverse document frequency and MOD) the increase in distance between centroids is more than the expansion of the cluster itself; whereas for the indexing methods that give less good search performance the relative sizes of the clusters are larger. However, the Q_1 and Q_2 values can not be depended upon to evaluate the search performance of a collection in place of recall and precision. The Q_1 and Q_2 values of the discrimination value model are larger than those of the inverse document frequency model, but from the search results in Table 9, the discrimination value model does not necessarily perform better than the inverse document frequency model. Similarly, the HDVD method performs worse than MODI(2), but the Q_1 value of MODI(2) for RCL

Precision Recall	Control	DIS	IDF	SKIP MOD (5, 50)	SKIP MOD (3, 38)	RCL MOD (5, 50)	HDVD	SKIP MODI (1)	SKIP MODI (2)	RCL MODI (1)	RCL MODI (2)
0.0	6468	6551	6551	6503	6437	6580	4834	5866	6041	5991	5994
0.05	6446	6551	6551	6503	6437	6558	4834	5866	6041	5969	5994
0.10	6361	6455	6415	6362	6390	6432	4698	5739	5931	5863	5834
0.15	5859	6045	6022	5943	5946	5977	4330	5188	5250	5296	5336
0.20	5696	5832	5845	5730	5622	5806	4160	4959	5053	5102	5174
0.25	5362	5488	5442	5430	5289	5423	3767	4698	4800	4724	4757
0.30	4930	5046	5001	4943	4744	4951	3161	4084	4255	4219	4328
0.35	4232	4275	4428	4354	4139	4403	2616	3290	3466	3542	3581
0.40	4054	4198	4327	4275	4028	4303	2559	3168	3362	3436	3437
0.45	3570	3747	3845	3766	3627	3813	2250	2854	3010	3073	3073
0.50	3546	3713	3829	3756	3603	3775	2243	2825	2988	3028	3025
0.55	3080	3236	3349	3213	3089	3209	1909	2501	2722	2588	2664
0.60	3012	3136	3252	3094	2965	3102	1859	2409	2603	2510	2581
0.65	2813	2914	3054	2880	2761	2810	1709	2261	2428	2378	2442
0.70	2330	2538	2573	2397	2298	2313	1413	1907	2017	2008	2050
0.75	2280	2478	2478	2321	2248	2240	1359	1832	1929	1921	1984
0.80	1883	2018	1992	1932	1867	1884	1190	1640	1607	1670	1623
0.85	1586	1775	1701	1641	1627	1666	0994	1294	1314	1357	1368
0.90	1502	1643	1542	1535	1512	1555	0936	1214	1249	1297	1274
0.95	1427	1565	1471	1480	1438	1481	0888	1154	1183	1250	1226
1.00	1427	1565	1471	1480	1438	1481	0888	1154	1183	1250	1226

Recall and Precision Values for the Full Searches of all the Indexing Methods

Table 9

clustered collection is smaller than that of the HDVD method. Furthermore, the difference in search performance between HDVD and control is more than that between the control and MOD method, but the differences in Q values are nearly the same. Nevertheless, the Q values do give a fairly compatible ranking of the performance of the various methods compared to the searches performed as in Table 9.

3. With the application of the HDVD method, the average correlation of documents to their centroids over all clusters does not vary for the relevant clustered collection. This is an indication that the deletion of high discrimination value terms, nearly one seventh of the concepts in the collection, has little effect within clusters. On the other hand, the sizes of the clusters vary with the use of the inverse skewness factor, as shown in MODI(1) and MODI(2) in Table 8. To a lesser extent the discrimination value model and the inverse document frequency model are also emphasizing the different effects of the terms. It becomes apparent that there are various functions played by the terms under different conditions, similar to the idea that there are terms that promote recall and others that promote precision. It is now obvious that terms are needed to distinguish documents that belong to different clusters; whereas within a cluster, the terms needed should have less discriminating power or should only discriminate documents within the cluster.

References

[1] C.J. Van Rijsbergen and K. Sparck Jones, A Test for the Separation of Relevant and Non-Relevant Documents in Experimental Retrieval Collections, Journal of Documentation, Vol. 29, No. 3, September 1973.

[2] K. Bonwit and J. Aste-Tonsmann, Negative Dictionaries, Scientific Report No. ISR-18 to the National Science Foundation and National Library of Medicine, Section VI, Department of Computer Science, Cornell University, October 1970.

[3] D. Murray, Document Retrieval Based on Clustered Files, Report No. ISR-20 to the National Science Foundation and National Library of Medicine, Department of Computer Science, Cornell University, Ithaca, June 1972.

[4] M. Kerchner, Dynamic Document Processing in Clustered Collections, Report ISR-19 to the National Science Foundation, Department of Computer Science, Cornell University, Ithaca, 1971.

[5] R. Williamson, Real-time Document Retrieval, Ph.D. Thesis, Cornell University, June 1974.

A Theory of Term Importance in Automatic Text Analysis

G. Salton*, C.S. Yang*, and C.T. Yu+

Abstract

Most existing automatic content analysis and indexing techniques are based on word frequency characteristics applied largely in an ad hoc manner. Contradictory requirements arise in this connection, in that terms exhibiting high occurrence frequencies in individual documents are often useful for high recall performance (to retrieve many relevant items), whereas terms with low frequency in the whole collection are useful for high precision (to reject nonrelevant items).

A new technique, known as discrimination value analysis ranks the text words in accordance with how well they are able to discriminate the documents of a collection from each other; that is, the value of a term depends on how much the average separation between individual documents changes when the given term is assigned for content identification. The best words are those which achieve the greatest separation.

The discrimination value analysis accounts for a number of important phenomena in the content analysis of natural language texts:

a) the role and importance of single words;

b) the role of juxtaposed words (phrases);

c) the role of word groups or classes, as specified in a thesaurus.

Effective criteria can be given for assigning each term to one of these three classes, and for constructing optimal indexing vocabularies.

* Department of Computer Science, Cornell University, Ithaca, N.Y. 14853

+ Department of Computer Science, University of Alberta, Edmonton, Alberta

The theory is validated by citing experimental results.

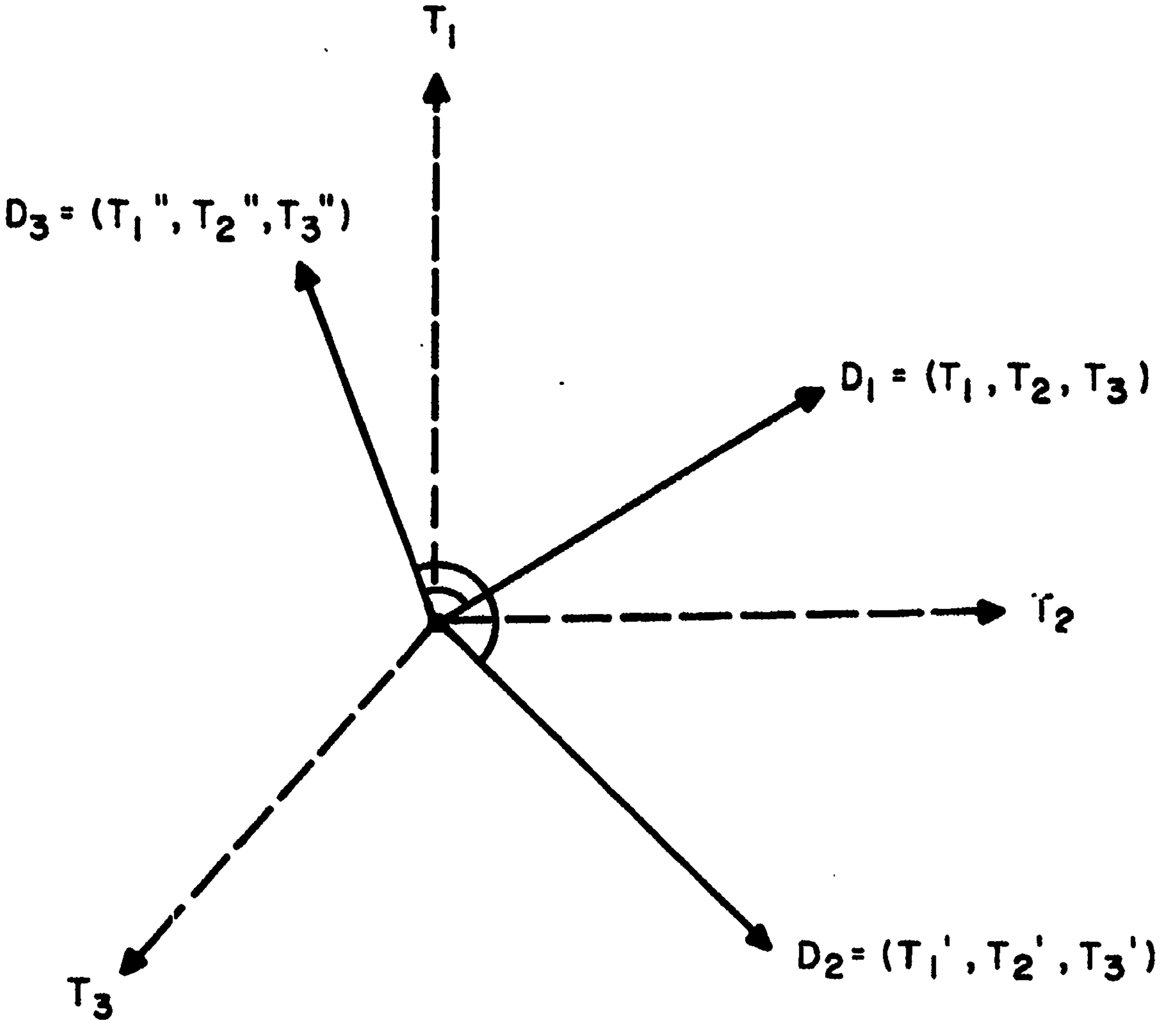
1. Document Space Configuration

Consider a collection of entities D (documents) represented by weighted properties w. In particular, let

$$D_i = (w_{i1}, w_{i2}, \ldots, w_{it}) \qquad (1)$$

where w_{ij} represents the weight of term j in the vector corresponding to the ith document. Given two documents D_i and D_j, it is possible to define a measure of relatedness $s(D_i, D_j)$ between the documents depending on the similarity of their respective term vectors. In three dimensions (when only three terms identify the documents), the situation may be represented by the configuration of Fig. 1, where the similarity between any two of the document vectors may be assumed to be a function inversely related to the angle between them. That is, when two document vectors are exactly the same, the corresponding vectors are superimposed and the angle between them is zero.

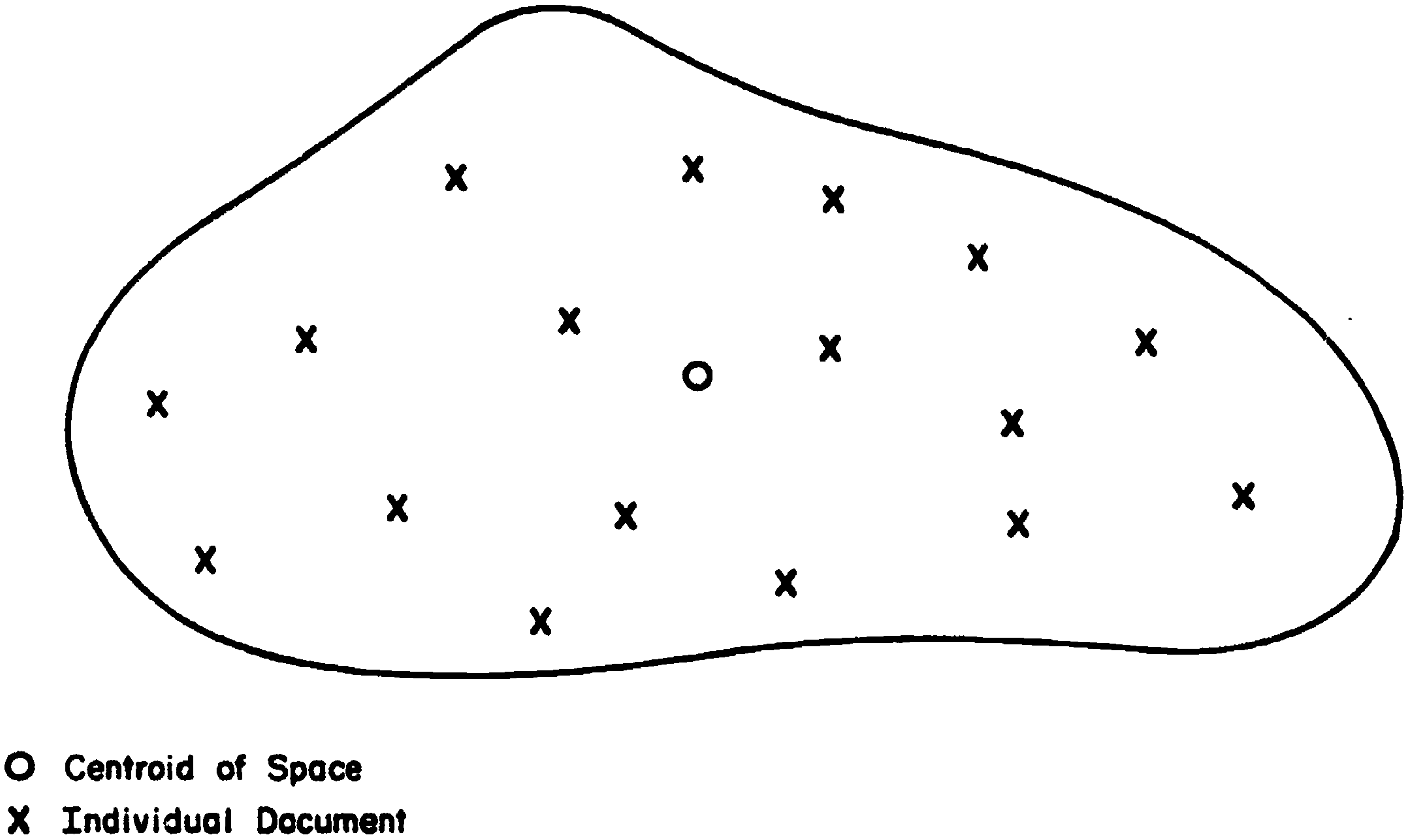
When the dimensionality of the space exceeds three, that is when more than three terms are used to identify a given document, the envelope of the vector space may be used to represent the collection as in the example of Fig. 2. Here only the tips of the document vectors are shown, represented by x's, and the distance between two x's is inversely related to the similarity between the corresponding document vectors — the smaller the distance between x's, the smaller will be the angle between the vectors, and thus the more similar the term assignments.



Vector Representation of

Document Space

Fig. 1



Multidimensional Document Space

Fig. 2

A central document, or centroid C, may be introduced, located in the center of the document space, which for certain purposes may represent the whole collection. The ith vector element c_i of the centroid can simply be defined as the average of the ith term w_{ij} across the n documents of the collection; that is

$$c_i = \frac{1}{n} \sum_{j=1}^{n} w_{ij}.$$

It is clear that a particular document space configuration, such as that of Fig. 2, reflects directly the details of the indexing chosen for the identification of the documents. This raises the question about the choice of an optimum indexing process, or alternatively, about an effective document space configuration. A number of studies, carried out over the last few years, indicate that a good document space is one which maximizes the average separation between pairs of documents. [1,2] In particular, the document space will be maximally separated, when the average distance between each document and the space centroid is maximized, that is, when

$$Q = \sum_{i=1}^{n} s(C, D_i) \qquad (2)$$

is minimum. Obviously, in such a case, it may be easy to retrieve each given document without also necessarily retrieving its neighbors. This insures a high precision output, since the retrieval of a given relevant item will then not also entail the retrieval of many nonrelevant items in its vicinity. Furthermore, when the relevant documents are located in the same general area of the space, high recall may also be obtainable, since many relevant items may then be correctly retrieved, and many nonrelevant correctly rejected.*

A particular indexing system, known as the discrimination value model assigns the highest weight, or value, to those terms which cause the maximum possible separation between the documents of a collection. This model is described and analyzed in the remainder of this study.

2. The Discrimination Value Model

The discrimination value of a term is a measure of the changes in space separation which occur when a given term is assigned to a collection of documents. A good discriminator is one which when assigned as an index term will render the documents less similar to each other; that is, its assignment decreases the space density. Contrariwise, a poor discriminator increases the density of the space. By computing the space densities both before and after assignment of each term, it is possible to rank the terms in decreasing order of their discrimination values.

In particular, consider a measure of the space density, such as the Q value given in equation (2), and let Q_k represent the density Q with the kth term removed from all document (and from the centroid) vectors. The discrimination value of term k may then be defined as

$$DV_k = Q_k - Q. \tag{3}$$

* Retrieval performance is often measured by parameters such as recall and precision, reflecting the ratio of relevant items actually retrieved, and of retrieved items actually relevant.

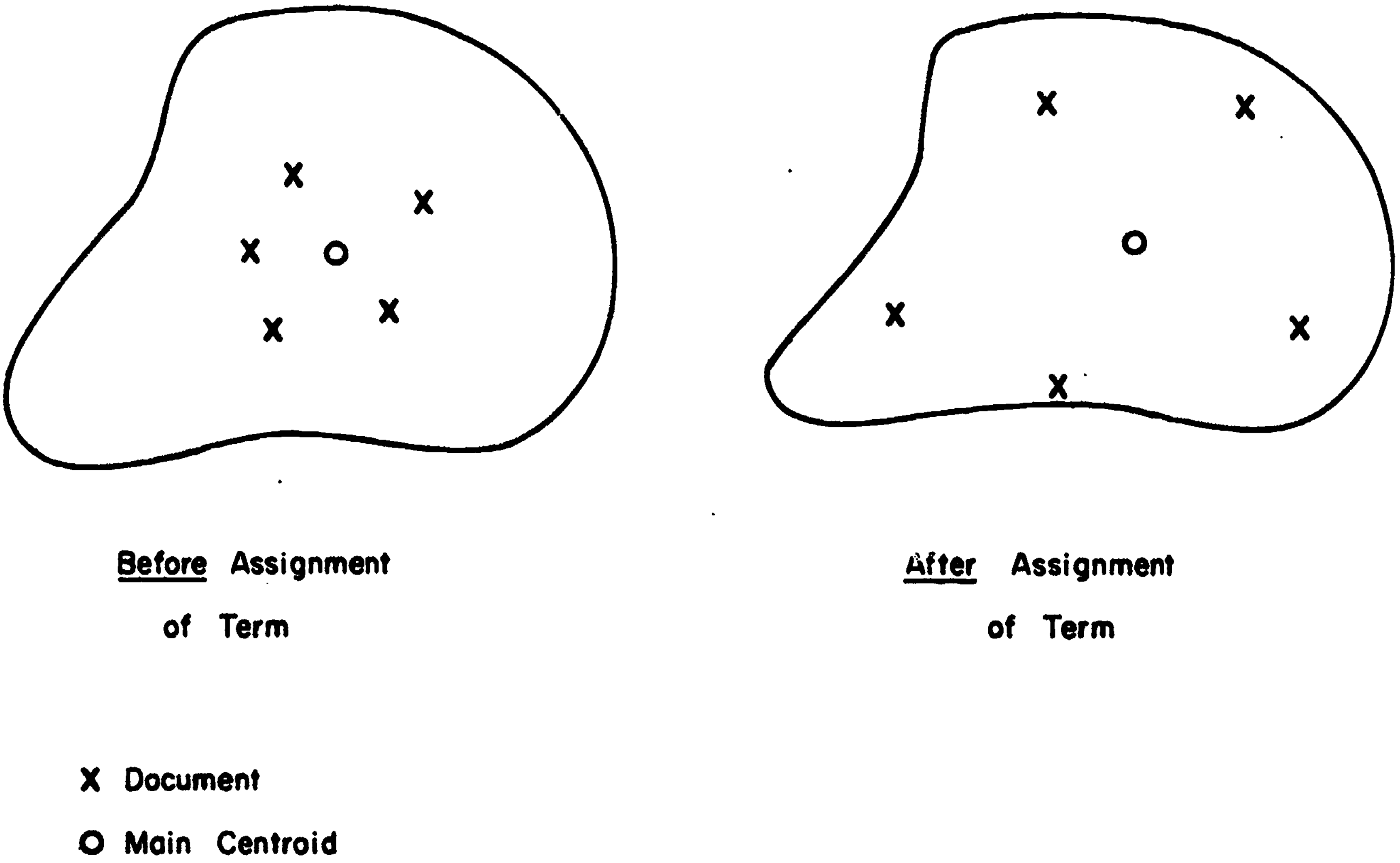
Obviously, if term Q is a good discriminator, then its removal will cause a compression in the document space (an increase in space density), because its assignment would have resulted in an increase in space separation. Thus for good discriminators $Q_k > Q$ and DV_k is positive. The reverse is true for poor discriminators whose removal causes a decrease in space density, leading to negative discrimination values. A vast majority of the terms may be expected to produce neither increase nor decrease in space density; in such a case a discrimination value near zero is obtained. The operations of a good discriminator are illustrated in the simplified drawing of Fig. 3.

In the retrieval experiments conducted earlier with three collections in aerodynamics (Cranfield collection, 424 documents comprising 2651 distinct terms), medicine (Medlars collection, 450 documents comprising 4726 terms), and world affairs (Time collection, 425 documents comprising 14098 terms), the discrimination value model produced excellent retrieval results. [1] In particular, a term weighting system which assigns to each term k a value w_{kj} consisting of the product of its frequency of occurrence in document j (f_{kj}) multiplied by its discrimination value DV_k,

$$w_{kj} = f_{kj} \cdot DV_k, \qquad (4)$$

produces recall and precision improvements of about ten percent over methods where only the term frequencies f_{kj} are taken into account.*

* Terms receiv.ng high weights according to expression (4) are those which exhibit high occurrence frequencies in certain specified documents, and at the same time can distinguish these documents from the remainder of the collection.



Before Assignment of Term

After Assignment of Term

X Document

O Main Centroid

Operation of Good Discriminating Term

Fig. 3

It may be of interest to inquire what kind of terms are favored by a weighting system such as that of expression (4), and what accounts for the value of the discrimination model. Some experimental evidence relating the discrimination values to certain frequency characteristics of the terms in the document collections is presented in the next section. This in turn, leads to an indexing theory to be examined in the remainder of this study.

3. Discrimination Values and Document Frequencies

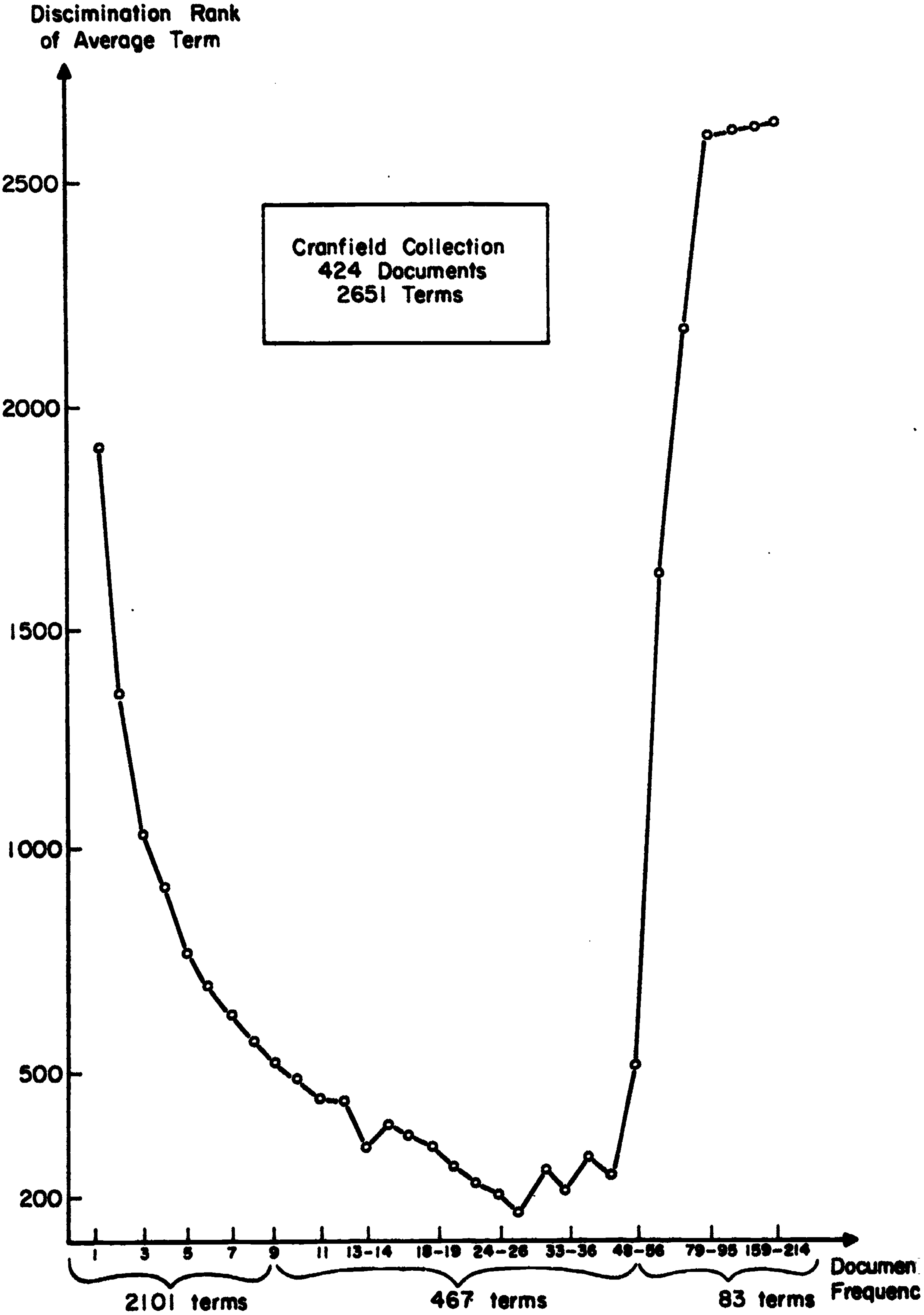
Consider any term k assigned to a collection of documents, and let d_k be its document frequency, defined as the number of documents in the collection to which term k is assigned. More specifically,

$$d_k = \sum_{j=1}^{n} b_{kj}$$

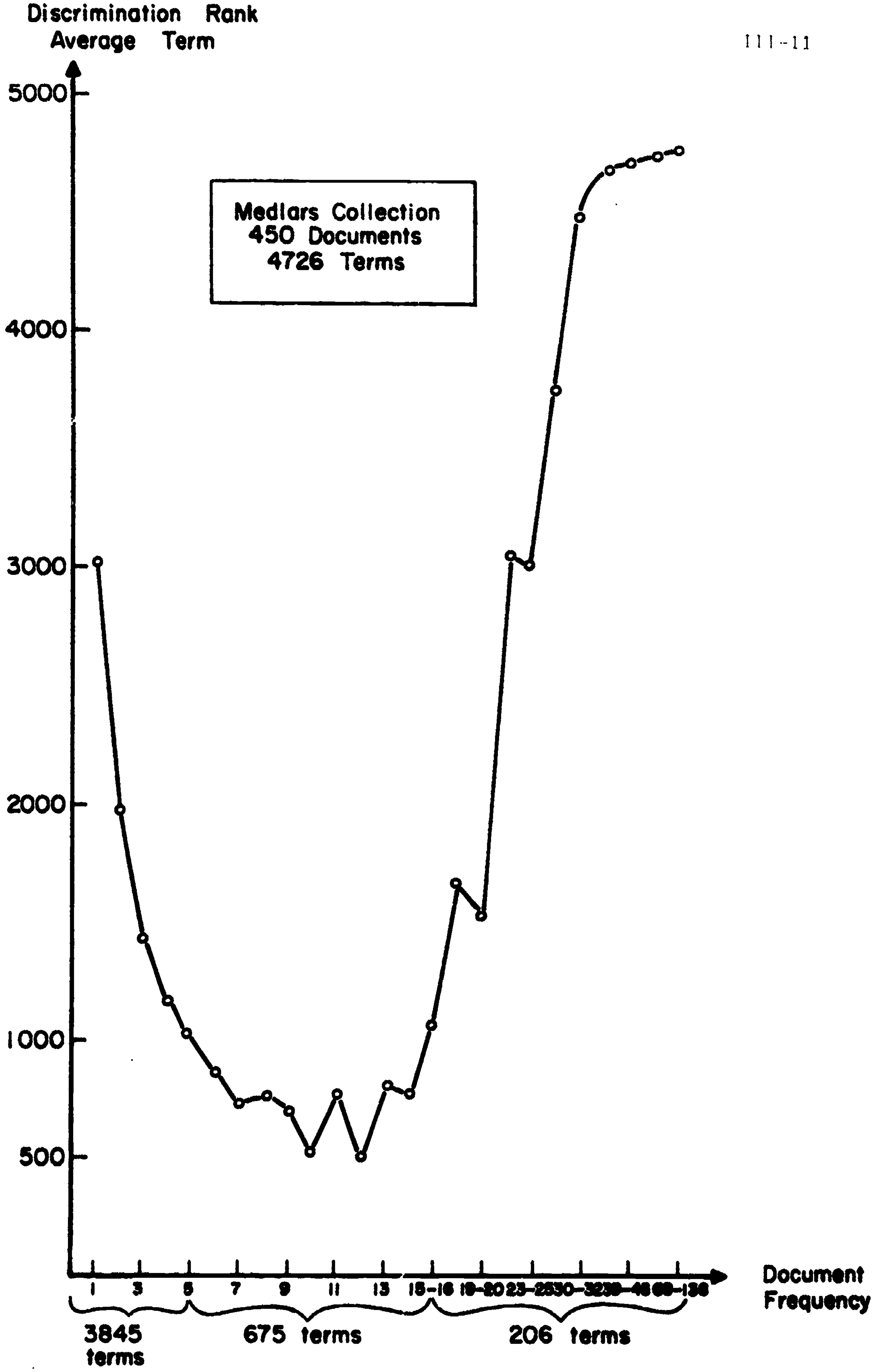
where $b_{kj} = 1$ whenever $f_{kj} \geq 1$, and $b_{kj} = 0$ otherwise. It is instructive to arrange the terms assigned to a document collection into disjoint sets in such a way that the terms assigned to a given set have equal document frequencies $d_k = I$. Moreover, for each such set of terms the average rank in decreasing discrimination value order may be computed, thereby relating document frequencies with discrimination values.*

A plot giving the average discrimination value rank for the terms exhibiting certain document frequency ranges is shown in Figs. 4(a), (b), and (c) for the collections in aerodynamics, medicine, and world affairs (Cranfield, Medlars, and Time) respectively. It may be seen that a U shaped curve is obtained in each case, with the following interpretation:

* For a set of t terms, the discrimination value rank ranges from 1 for the best discriminator to t for the worst.



Average Discrimination Value Rank of Terms
Fig. 4(a)



Average Discrimination Value Rank of Terms

Fig. 4(b)

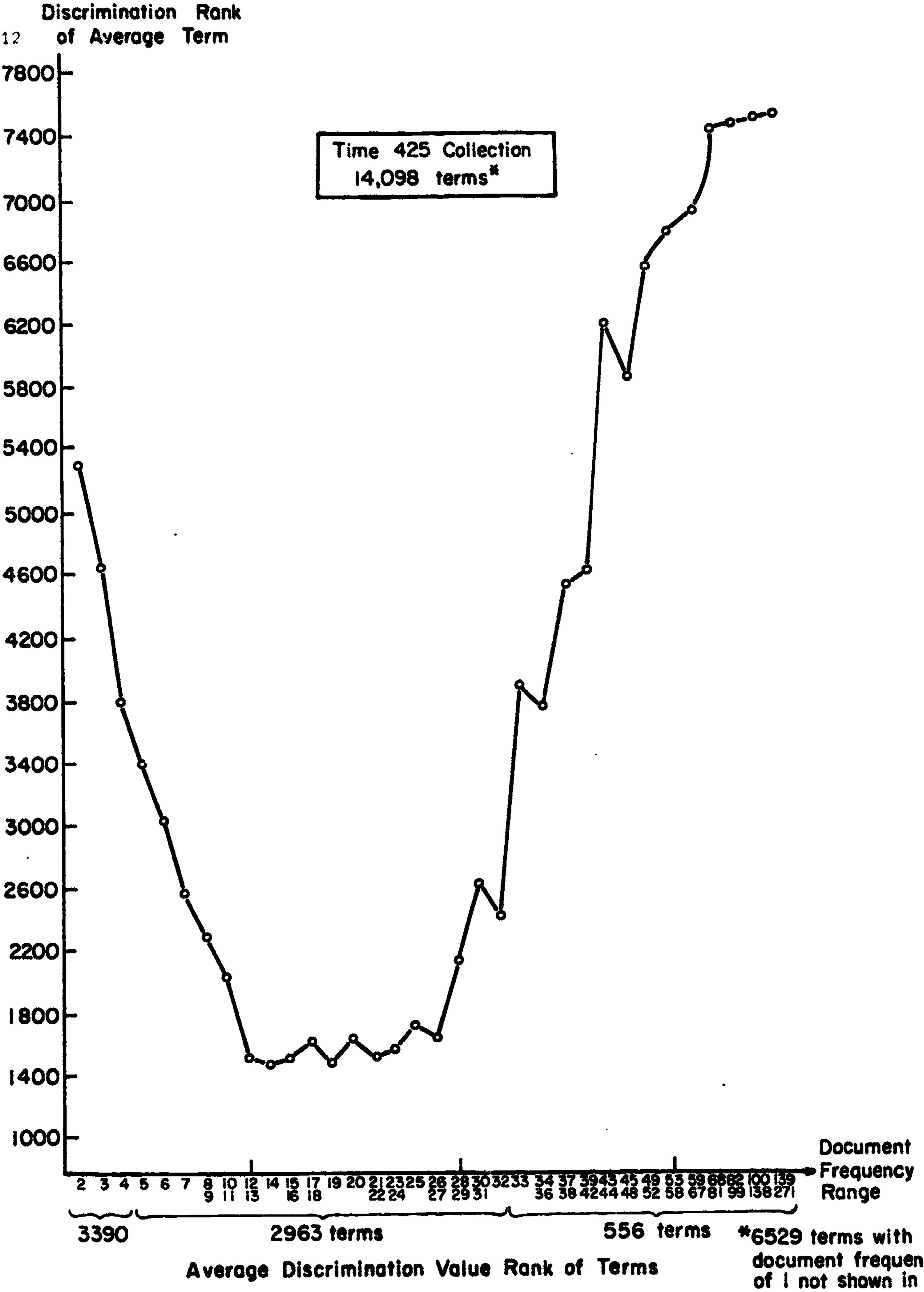


Fig. 4(c)

*6529 terms with document frequency of 1 not shown in figure.

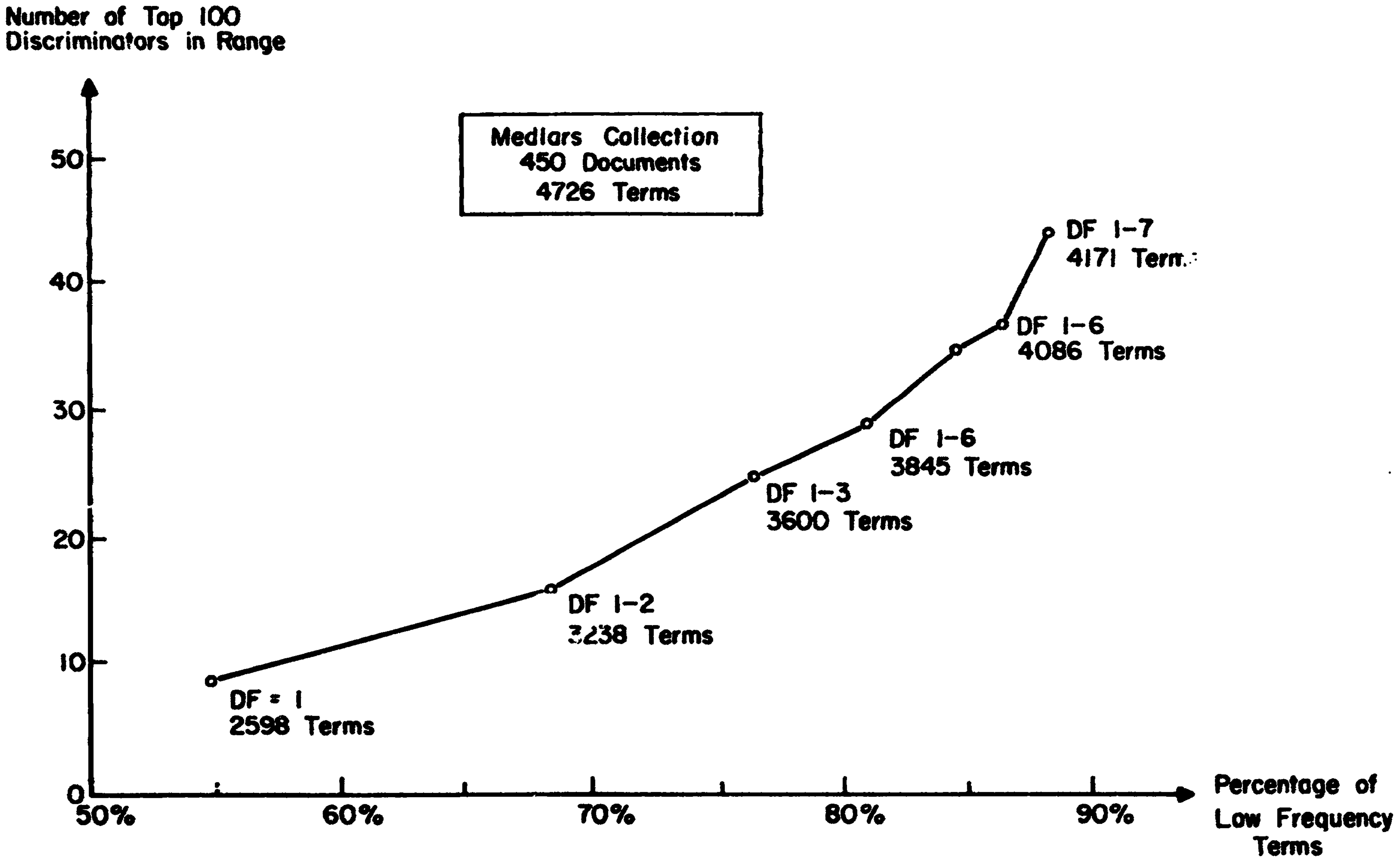
a) the terms with very low document frequencies, located on the left-hand side of Fig. 4 are poor discriminators, which average discrimination value ranks in excess of t/2 for t terms;

b) the terms with high document frequencies exceeding n/10, located on the right-hand side of Fig. 4 are the worst discriminators, with average discrimination value ranks near t;

c) the best discriminators are those whose document frequency is neither too high nor too low — with document frequencies between n/100 and n/10 for n documents; their average discrimination value ranks are generally below t/5.

The output of Fig. 4 shows average discrimination value ranks only. Before deciding that all terms with low and high document frequencies can automatically be disregarded, it is useful to determine whether any good discriminators are in fact included in the corresponding low frequency and high frequency term sets. Figs. 5(a) and 5(b) show sets of low frequency terms for the Medlars and Time collections respectively, together with the number of good discriminators — those with discrimination ranks between 1 and 100 — included in each set. Fig. 5 shows overlapping term sets, consisting of all terms with document frequency equal to 1, 1 and 2, 1 to 3, etc., together with the percentage figures of the total number of terms represented by the corresponding sets.

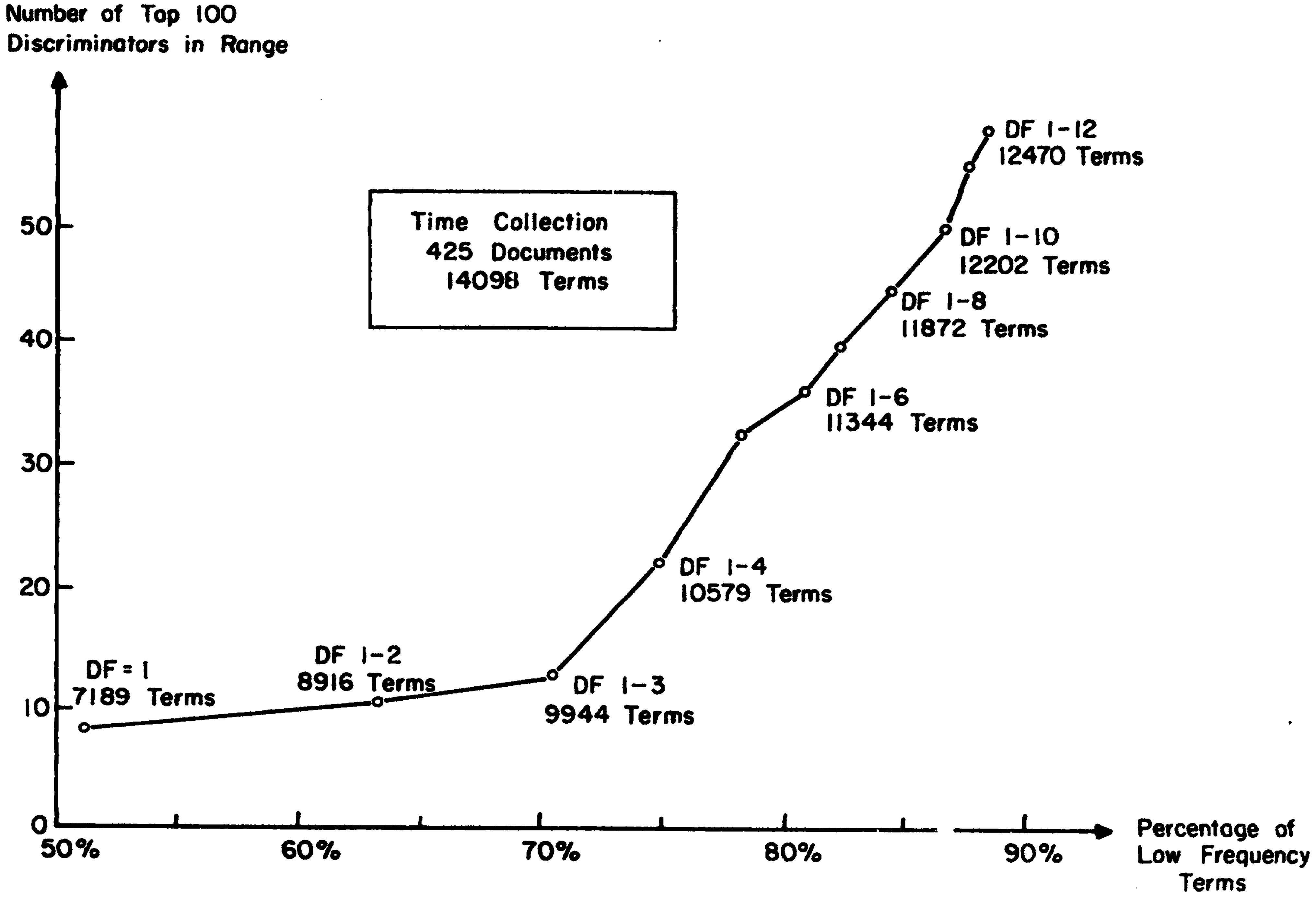
Thus when seventy percent of the terms are taken in increasing document frequency order — corresponding in the Medlars collection to about 3200 terms out of 4700 with document frequencies of 1 or 2, and in the Time collection to 9900 terms out of 14000 with document frequencies 1 to 3 — it is seen that only about 15 good discriminators are included for Medlars, and about 12 for Time. When the proportion of terms increases to eighty percent in increasing document frequency order, including 3800 Medlars terms, or 11300 Time terms,



Number of Good Discriminators Among

Low Frequency Terms

Fig. 5(a)



Number of Good Discriminators Among Low Frequency Terms

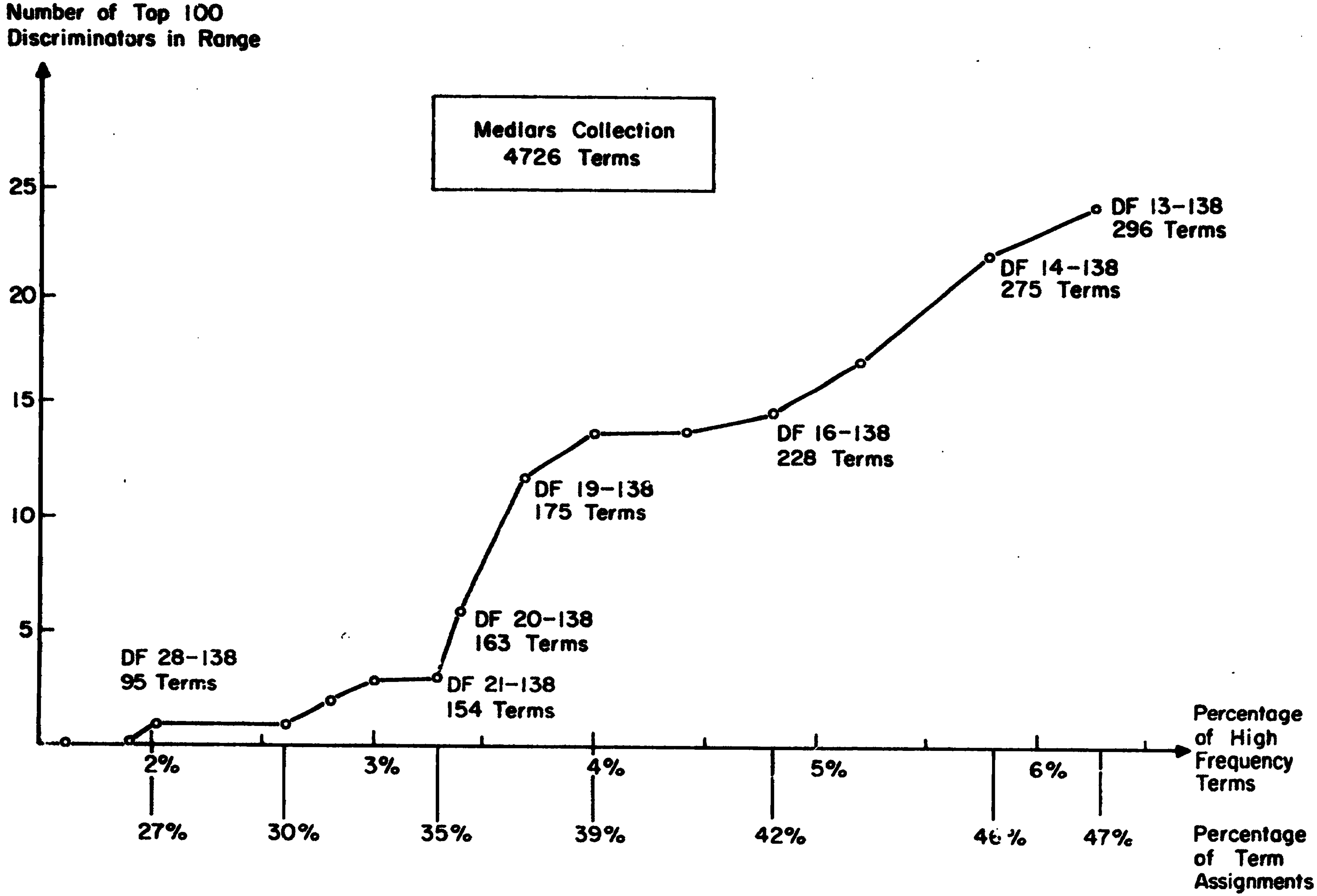
Fig. 5(b)

ranging in document frequency from 1 to 6, the number of good discriminators rises to 30 for Medlars and 35 for Time. When so few good terms are included among the mass of low frequency terms, it is obvious that special provisions must be made in any indexing process for the utilization of these terms.

Consider now the very high-frequency terms — those which according to the output of Fig. 4 exhibit the lowest discrimination values. While the number of such terms is not large, each of the terms accounts for a substantial portion of the total term assignments to the documents of a collection because of the high document frequency involved.

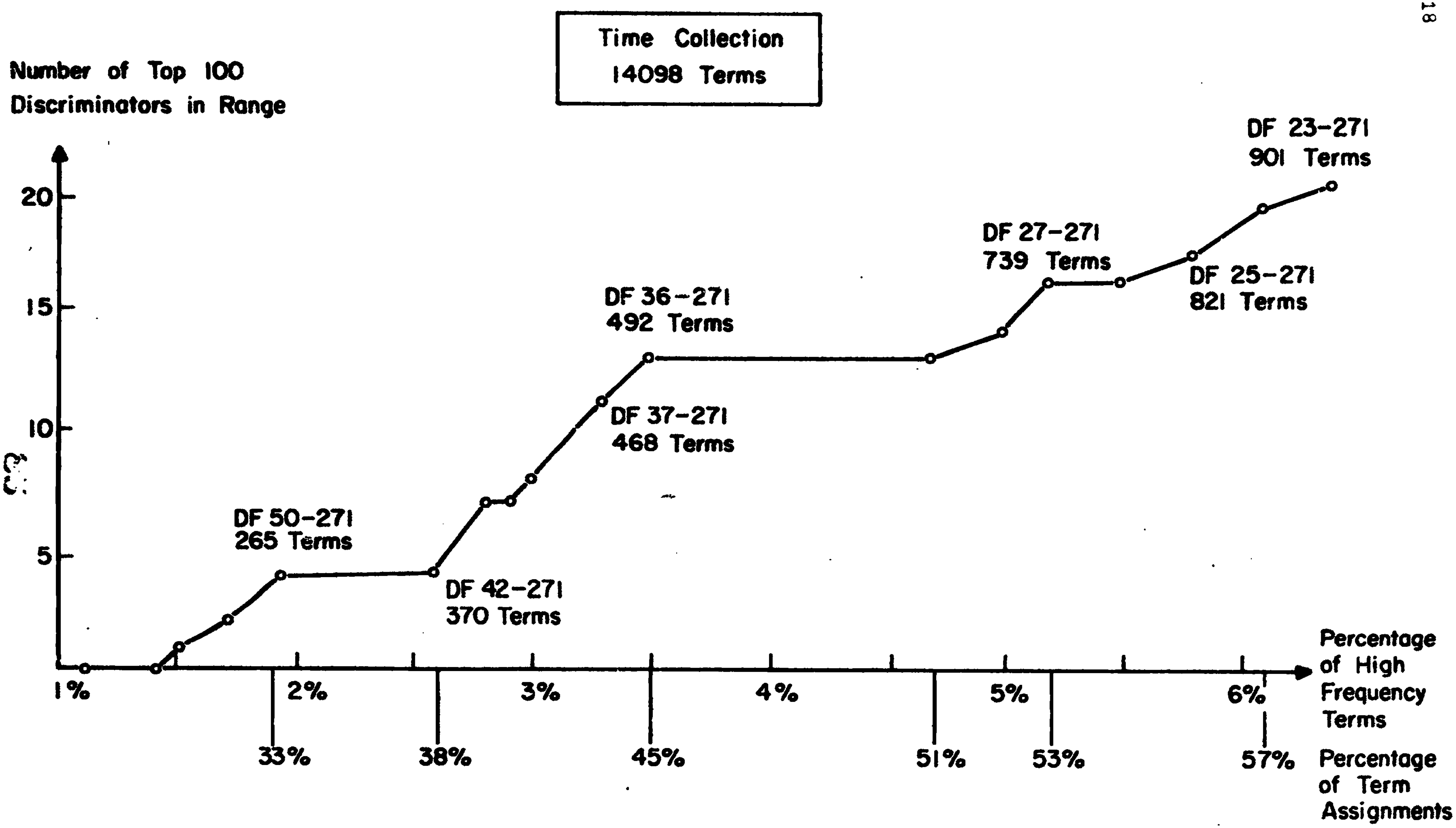
The output of Fig. 6(a) for Medlars, and 6(b) for Time shows that about four percen of the high-frequency terms present in a document collection, accounts for forty to fifty percent of all term assignments, when the terms are taken in decreasing document frequency order. The absolute number of distinct terms is 200 approximately for the Medlars collection and about 500 for Time. In each case, less than 15 of these terms are classified as good discriminators. When the proportion of terms taken in high frequency order increases to six percent, accounting for 46 percent of the term assignments in Medlars, and 57 percent for Time, the number of good discriminators increases to about 20 in each case.

The information included in Figs. 5 and 6 is summarized in Table 1. In each case, certain cutoff percentages are given for terms taken either in low document frequency or in high document frequency order. For each such percentage, the number of good discriminators included in the corresponding term set is stated for each of the three test collections. Thus when sixty percent of the terms are taken in increasing document frequency order, not



Number of Good Discriminators Among High Frequency Terms

Fig. 6(a)



Number of Good Discriminators Among High-Frequency Terms

Fig. 6(b)

Type of Terms	Fraction of Terms Covered (Fraction of Term Assignments)	Number of Terms	Document Frequency of Terms	Number of Good Discriminators	
				From Top 50	From Top 100
Low Frequency	60%	CRAN 1668	1-3	0	0
		MED 3238	1-2	5	16
		TIME 8916	1-2	1	11
	70%	CRAN 1899	1-4	0	4
		MED 3600	1-3	8	25
		TIME 9944	1-3	2	13
	80%	CRAN 2153	1-9	5	14
		MED 3845	1-4	8	29
		TIME 11344	1-6	12	36
High Frequency	(42%)	CRAN 91	54-214	5	11
	3.5% (36%)	MED 163	20-138	5	6
	(45%)	TIME 492	36-271	8	13
	(45%)	CRAN 105	48-214	7	15
	4.0% (39%)	MED 192	18-138	10	14
	(47%)	TIME 555	33-271	8	13
	(47%)	CRAN 115	44-214	9	17
	4.5% (40%)	MED 206	17-138	10	14
	(51%)	TIME 660	29-271	8	13

Number of Good Discriminators for Various Deletion
Percentage of Low and High Frequency Terms
Table 1

a single good discriminator is included among the 1668 terms for the Cranfield collection; only 5 of the top 50 terms, or 16 of the top 100, are present among the 3238 Medlars terms; finally, for Time 1, out of the top 50, or 11 of the top 100 are included among the first 8916 low frequency terms.

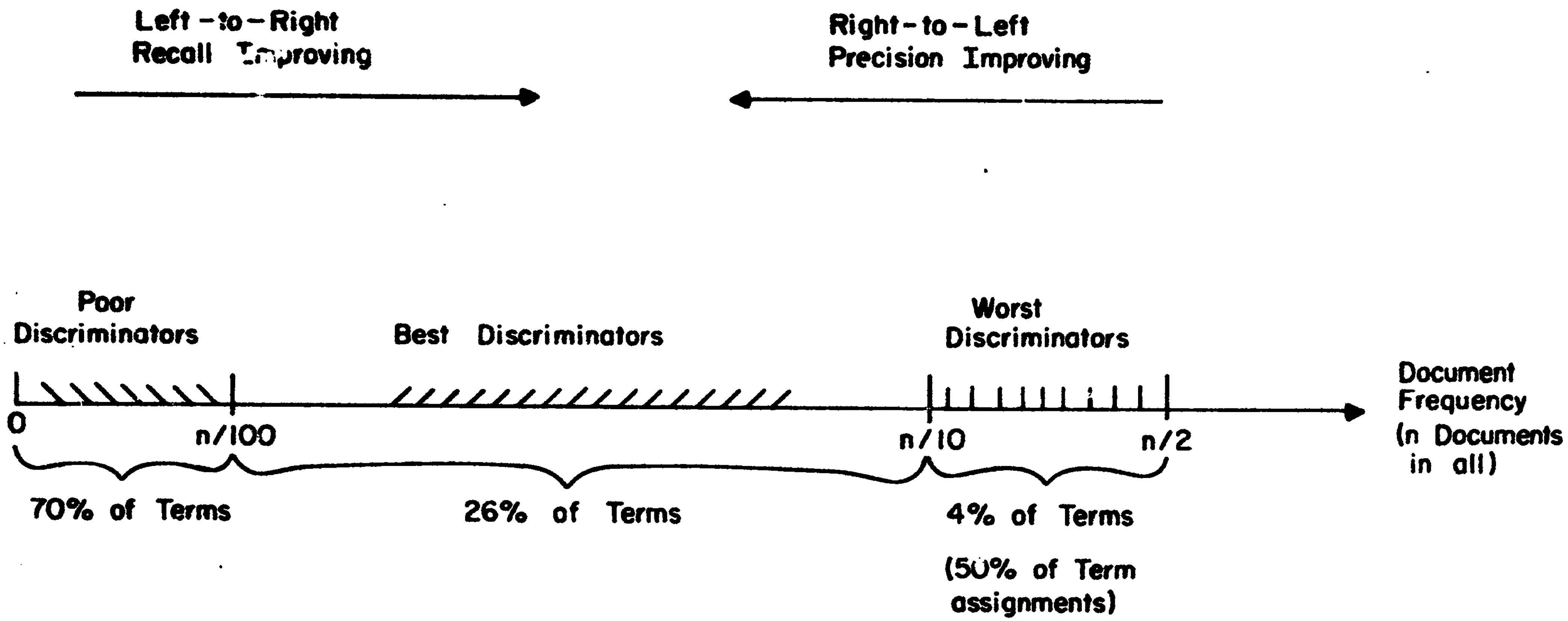
The number of good discriminators included among the high frequency terms for the three collections is similarly low, as shown in the bottom half of Table 1.

The conclusion to be reached from the data of Figs. 5 and 6 and of Table 1 is that very few good discriminators are included among the bottom seventy percent, or among the top four percent when the terms included in a collection of documents are taken in increasing document frequency order. This fact is used to construct an indexing strategy in the remainder of this study.

4. A Strategy for Automatic Indexing

Consider the graph of Fig. 7 in which the terms are once again arranged in increasing document freauency order. If the assumption is correct that the best terms for indexing purposes are concentrated in the set whose document frequency is neither too high nor too low — the frequency being approximately between n/100 and n/10 — then the following term transformations should be undertaken:

a) Terms whose document frequency lies between n/100 and n/10 should be used for indexing purposes directly without any transformation; these terms include the vast majority of the good discriminators.



Summarization of Discrimination Value of

Terms in Frequency Ranges

Fig. 7

b) Terms whose document frequency is too high — above n/10 — comprise the worst discriminators. These terms are too general in nature, or too broad, to permit proper discrimination among the documents; hence their use produces an unacceptable precision loss (it leads to the retrieval of too many items that are extraneous). These terms should be transformed into lower frequency terms — right-to-left on the graph of Fig. 7 — thereby enhancing the precision performance.

c) Terms whose document frequency is too low — below n/100 — are so rare and specific that they cannot retrieve an acceptable proportion of the documents relevant to a given query; hence their use depresses the recall performance. These terms should be transformed into higher frequency terms — left-to-right on the graph of Fig. 7 — thereby enhancing the recall performance.

It remains to describe the right-to-left and left-to-right transformations that may be used to generate useful indexing vocabularies. The obvious way of transforming the high frequency terms into lower frequency entities is to combine them into indexing phrases. In general, a phrase such as "programming language" exhibits a lower assignment frequency than either of the high frequency components "language" or "program". The summary of Fig. 7 then indicates that:

> Indexing phrases should be constructed from high frequency single term components in order to enhance the precision performance of the retrieval system.

The other left-to-right transformation which is required for recall enhancing purposes is now equally obvious. Low frequency terms with somewhat similar properties, or meanings, can be combined into term classes, normally specified by a thesaurus of related terms, or synonym dictionary. When a single term is replaced for indexing purposes by a thesaurus class consisting of several terms, the assignment frequency of the thesaurus class will in general exceed that of any of the components included in the class. Thus:

> The main virtue of a thesaurus is its ability to group a number of low frequency terms into thesaurus classes, thereby enhancing the recall performance.

A large number of different strategies is available for the generation of indexing phrases and term thesauruses. Consider first the criteria used for the formation of phrases. A phrase might be created whenever two or more components cooccur in the same document, or query; or when they cooccur in the same paragraph, or sentence of a document; or when they occur in certain specified positions within the same sentences; or, finally, when they cooccur in certain specified positions in a text while exhibiting certain predetermined syntactical relationships. The methods needed to identify the indexing phrases attached to a given document or query may then range from quite simple (any pair of noncommon terms cooccurring in a document may represent a phrase) to quite complex (the various phrase components must exhibit appropriate syntactical relationships and these relationships must be ascertained). [3]

For present purposes, a compromise position is adopted which bypasses an expensive syntactic analysis system in favor of the following procedure:

a) phrases are defined by using query texts;

b common function words are removed and a suffix deletion method is used to reduce the remaining query words to word stems;

c) the remaining word stems are taken in pairs, and each pair defines a phrase provided that the distance in the text between the two phrase components does not exceed two (at most one intervening word occurs between components), and provided that at least one of the components of each phrase is a high-frequency term;

d) phrases for which both components are identical are eliminated;

e) duplicate phrases, where all components match an already existing phrase are elminated.

The texts of all documents are checked for the presence of any phrase thus defined from the query statements, and appropriate weights are assigned.

The phrase formation process is illustrated in Fig. 8 for a query dealing with world affairs. It is seen that this query gives rise to eight distinct phrases with adjacent components, plus seven additional phrases for which the components are separated by one intervening word in the reduced query text.

It remains to determine an appropriate weight to be assigned to each phrase created by the foregoing process. Thus if terms p and q exhibit weights w_{ip} and w_{iq}, respectively in document i, corresponding, for example to the frequencies of occurrence of the respective terms in the document, the phrase consisting of components p and q might be assigned weight w_{ipq} defined as

$$w_{ipq} = \frac{w_{ip} + w_{iq}}{2} . \qquad (5)$$

A somewhat more refined weighting method uses w_{ipq} in conjunction with an "inverse document frequency" (IDF) factor which gives higher weights to phrases that occur comparatively rarely in the collection. The original inverse document frequency (IDF) factor, introduced by Sparck Jones, was defined as [4]:

$$IDF_k = \lceil \log_2 n \rceil - \lceil \log_2 d_k \rceil + 1,$$

QUERY: COALITION GOVERNMENT TO BE FORMED IN ITALY BY THE LEFT-WING SOCIALISTS, THE REPUBLICANS, SOCIAL DEMOCRATS, AND CHRISTIAN DEMOCRATS.

DELETE COMMON WORDS AND ELIMINATE SUFFIXES:

COALI GOVERN FORM ITALY LEFT-W SOCIAL REPUBLIC
SOCIAL DEMOCRAT CHRIST DEMOCRAT

PHRASES:

ADJACENT COMPONENTS	ONE INTERVENING WORD
COALI GOVERN, GOVERN FORM,	COALI FORM, GOVERN ITALY,
FORM ITALY, ITALY LEFT-W,	FORM LEFT-W, ITALY SOCIAL,
LEFT-W SOCIAL, SOCIAL REPUBLIC,	LEFT-W REPUBLIC, ~~SOCIAL SOCIAL~~,+
~~REPUBLIC SOCIAL~~*, SOCIAL DEMOCRAT,	REPUBLIC DEMOCRAT,
DEMOCRAT CHRIST. ~~CHRIST DEMOCRAT~~*	SOCIAL CHRIST, ~~DEMOCRAT DEMOCRAT~~+

* Duplicate Phrases Eliminated

+ Identical Components Eliminated
Word Order Immaterial

Sample Phrase Formation Process

Fig. 8

where IDF_k is the IDF factor for term k, and d_k is the document frequency of term k in a collection of n documents. Clearly IDF_k is large when d_k is small, and becomes small as d_k approaches n.

By analogy, a phrase IDF factor may be defined as:

$$IDF_{pq} = (\log n - \frac{\log d_p + \log d_q}{2}), \qquad (6)$$

where d_p and d_q are the respective document frequencies of phrase components p and q.

In conformity with the composite weighting system of equation (4) which uses the product of term frequencies and discrimination values, a composite phrase weight $\underline{W}_{ipq}$ for phrase pq in document i may then be defined as the product of the IDF factor and the average component weight (equations (5) and (6)):

$$\underline{W}_{ipq} = \left[\log n - \frac{\log d_p + \log d_q}{2}\right] \cdot \left[\frac{w_{ip} + w_{iq}}{2}\right] .* \quad (7)$$

In a retrieval environment, the phrases defined by the foregoing pre-edure may be used to replace the original phrase components — that is, the original components may be removed from the document and query vectors before the phrase identifiers are added. Alternatively, phrase components may be used in addition to the single term components. For the experiments described in the next section, the former policy was used in that phrases are introduced replacing the original component terms.

Consider now the converse to the right-to-left phrase formation process, namely the left-to-right thesaurus construction method. Here

* As before, the weighting system of expression (7) assigns high weights to phrases with highly weighted components in individual documents but with relatively low overall document frequency in the collection.

the notion is to use low frequency terms and to assemble them into classes of terms replacing the original vector components. If d_p and d_q are the document frequencies of terms p and q respectively, the document frequency of the class which includes both p and q may be defined as

$$D_{pq} = d_p + d_q - d_{pq}$$

term q, and both p and q, respectively. In general D_{pq} may be expected to be larger than either d_p or d_q individually. When m terms are included in a given term class, the document frequency of the class is defined simply as the number of documents in which at least one term assigned to that class appears.

Term classes are often defined by a thesaurus, and a given thesaurus class normally includes terms that are sufficiently similar in meaning, or context, to make it reasonable to ignore their differences for indexing purposes. A great many thesaurus construction procedures have been described in the literature including manual term grouping as well as fully automatic methods. [5,6,7,8] Among the latter are the so-called associative indexing procedures, where statistically associated terms are jointly assigned to the documents of a collection, and a variety of term clustering methods designed to group into a common class those terms which exhibit similar term assignments to the documents of a collection.

For experimental purposes it may be sufficient to use existing manually constructed thesauruses for the three test collections, and restricting the thesaurus to include only classes whose document frequency does not exceed a stated maximum. Such a thesaurus then effectively limits the number of high-frequency terms than can appear in any class, and provides the left-to-right frequency transformation specified by the model of Fig. 7. The weight with which a thesaurus class is assigned to a document or query vector may be defined as the average weight of the component terms originally present in that vector.

A frequency-restricted thesaurus such as the one described above may not specify classes that are completely identical with the term classes obtainable by initially using only the low frequency terms for a separate term clustering process; however the experimental recall-precision results may be expected to be close to those produced by an original thesaurus construction method.

The recall-precision results obtained from the operations modelled in Fig. 7 are examined in the next section.

5. Experimental Results

The right-to-left phrase formation process is designed to produce lower frequency entities from high frequency components, and vice versa for the left-to-right thesaurus grouping process. The data of Table 2 prove that the required frequency alterations are in fact obtained by the two transformations for the test collections in use.

Table 2(a) shows that the document frequency of the phrases is only about one third as large as the frequency of the individual components entering the phrase formation process. In Table 2(b) the reverse is seen to be the case for the thesaurus concepts whose document frequency is one and a half times that of the individual thesaurus entries. If the model of Fig. 7 specifying ideal frequency characteristics for index terms is appropriate, considerably better recall and precision output should be obtainable with the transformed terms (phrases and thesaurus classes) than the originals.

	Minimum Document Frequency needed for High-Frequency Component	Average Document Frequency	
		Single Terms Entering Phrase Process	Phrases
CRANFIELD	(45)	106	39
MEDLARS	(22)	40	7
TIME	(49)	101	38

Average Document Frequency for Phrases

Table 2(a)

	Maximum Document Frequency needed for Thesaurus Class to Insure Inclusion	Average Document Frequency	
		Single Terms Entering Thesaurus Process	Thesaurus Classes
CRANFIELD	(60)	24	32
MEDLARS	(40)	10	16
TIME	(60)	17	31

Average Document Frequency for Thesaurus Classes

Table 2(b)

Detailed recall-precision output is contained in Tables 3 and 4, and in the summary in Table 5 for the various indexing methods applied to the three test collections in aerodynamics, medicine, and world affairs. Performance figures comparing the standard term frequency weighting (f_{ki}) for single terms k in documents i with the phrase process are shown in Table 3. The phrase procedure uses the normal single terms in addition to indexing phrases weighted in accordance with the formula of expression (7).

Table 3 contains precision figures averaged over 24 user queries for each of the test collections at ten specified recall levels ranging in magnitude from 0.1 to 1.0 in steps of 0.1. The percentage improvement in precision for the phrase process over the standard is also given at each recall level, together with an average improvement ranging from a high of 39 percent for the Medlars collection to a low of 17 percent for Time.

Table 4 contains output similar to that already shown in Table 3. However the data in Table 4 apply to an indexing system using both left-to-right (thesaurus) and right-to-left (phrase) transformations. It is seen from Table 4 that the thesaurus transformation adds an additional average improvement of 13 percent in precision for the Medlars collection; additional advantages are also obtained for the Cranfield and Time collections.

The evaluation results are summarized in Table 5. It is seen that average precision values of approximately 0.70, 0.40, and 0.20 at high, medium, and low precision are transformed into average figures of 0.90, 0.60 and 0.30 approximately when the discrimination properties of the terms are optimized. The retrieval results displayed in Tables 3, 4, and 5 have not been surpassed by any manual or automatic indexing procedures previously

	CRAN 424			MED 450			TIME 425		
	Standard Term Frequency	Phrase Assignment	Advantage	Standard Term Frequency	Phrase Assignment	Advantage	Standard Term Frequency	Phrase Assignment	Advantage
.1	.6844	.8793	+28%	.7891	.8811	+12%	.7496	.8483	+13%
.2	.5303	.7344	+38%	.6730	.8149	+21%	.7071	.8419	+19%
.3	.4683	.6013	+28%	.5481	.6992	+28%	.6710	.7998	+19%
.4	.3482	.5205	+49%	.4807	.6481	+35%	.6452	.7729	+20%
.5	.3134	.4150	+32%	.4384	.5930	+35%	.6351	.7025	+11%
.6	.2556	.3623	+42%	.3721	.5450	+46%	.5866	.6800	+16%
.7	.1989	.3017	+52%	.3357	.4867	+45%	.5413	.6331	+17%
.8	.1631	.1953	+20%	.2195	.3263	+49%	.5081	.5895	+16%
.9	.1265	.1463	+16%	.1768	.2767	+56%	.3865	.4618	+19%
1.0	.1176	.1314	+12%	.1230	.1959	+60%	.3721	.4529	+22%
		Average	+32%		Average	+39%		Average	+17%

Average Precision Values at Ten Recall Points

(Phrase Process vs. Standard)

Table 3

	CRAN 424			MED 450			TIME 425		
	Standard Term Frequency	Thesaurus Plus Phrases	Advantage	Standard Term Frequency	Thesaurus Plus Phrases	Advantage	Standard Term Frequency	Thesaurus Plus Phrases	Advantage
.1	.6844	.8745	27.8%	.7891	.8919	13.0%	.7496	.8339	11.2%
.2	.5305	.7905	49.1%	.6750	.8331	23.4%	.7071	.8133	15.0%
.3	.4689	.6387	36.2%	.5481	.7057	28.8%	.6740	.7812	16.4%
.4	.3482	.5401	55.1%	.4807	.6443	34.0%	.6452	.7681	19.0%
.5	.3134	.4516	44.1%	.4384	.6099	39.1%	.6351	.7006	10.3%
.6	.2556	.3718	45.5%	.3721	.5548	49.1%	.5869	.6882	17.3%
.7	.1989	.2979	49.8%	.3357	.5179	54.3%	.5413	.6389	18.0%
.8	.1631	.2019	23.8%	.2195	.3949	79.9%	.5004	.5915	18.2%
.9	.1265	.1556	23.0%	.1768	.3505	98.2%	.3865	.4842	25.3%
1.0	.1176	.1375	16.9%	.1230	.2494	101.9%	.3721	.4790	28.7%
		Average	+37%		Average	+52%		Average	+18%
		Average (Phrases)	+32%		Average (Phrases)	+39%		Average (Phrases)	+17%
			+ 5%			+13%			+ 1%

Average Precision Values at Ten Recall Points

(Thesaurus and Phrases vs. Standard)

Table 4

CRAN 424	MED 450	TIME 425
Automatic Phrases vs. Standard Term Frequency +32%	Automatic Phrases vs. Standard Term Frequency +39%	Automatic Phrases vs. Standard Term Frequency +17%
Automatic Phrases Plus Thesaurus vs. Standard Run +37%	Automatic Phrases Plus Thesaurus vs. Standard Run +52%	Automatic Phrases Plus Thesaurus vs. Standard Run +18%
Best Precision Low Recall 0.89 Medium Recall 0.43 High Recall 0.13	Best Precision Low Recall 0.88 Medium Recall 0.61 High Recall 0.23	Best Precision Low Recall 0.85 Medium Recall 0.70 High Recall 0.45

Summary of Recall-Precision Evaluation (Three Collections)

Table 5

tried with sample document collections and user queries. Furthermore, because of the high average precision values produced by the indexing theories described in this study, it is not likely that additional drastic improvements in retrieval effectiveness are obtainable in the foreseeable furture.

References

[1] G. Salton and C.S. Yang, On the Specification of Term Values in Automatic Indexing, Journal of Documentation, Vol. 29, No. 4, December 1973, p. 351-372.

[2] G. Salton, A. Wong, and C.S. Yang, A Vector Space Model for Automatic Indexing, Technical Report No. 74-208, Department of Computer Science, Cornell University, Ithaca, N.Y., July 1974.

[3] G. Salton and M.E. Lesk, Computer Evaluation of Indexing and Text Processing, Journal of the ACM, Vol. 15, No. 1, January 1968, p. 8-36.

[4] K. Sparck Jones, A Statistical Interpretation of Term Specificity and its Application to Retrieval, Journal of Documentation, Vol. 28, No. 1, March 1972, p. 11-20.

[5] K. Sparck Jones, Automatic Keyword Classifications, Butterworths, London, 1971.

[6] C.C. Gotlieb and S. Kumar, Semantic Clustering of Index Terms, ACM Journal, Vol. 15, No. 4, October 1968, p. 493-513.

[7] G. Salton, Experiments in Automatic Thesaurus Construction for Information Retrieval, Information Processing 71, North Holland Publishing Co., Amsterdam, 1972, p. 115-123.

[8] G. Salton, C.S. Yang, and C.T. Yu, Contributions to the Theory of Indexing, Proc. IFIP Congress-74, Stockholm, August 1974.

Negative Dictionary Construction

R. Crawford

1. Introduction

Effective information retrieval is based on the ability to provide an accurate description of each item and to be able to discriminate between the available information items. In the area of document retrieval, a set of words (terms) chosen from the subject area of the documents may be used to describe the documents. [1,2,3] If the set of words used to describe each document is chosen properly, then each document will have a description which is both accurate and unique in relation to the other documents. The document descriptions should reflect the same differences and similarities between documents as would be noticed by a reader of the original documents.

Thus, for a collection of documents in a particular subject area, two problems are apparent. First, a set of terms must be chosen for use in describing the documents in the collection. This set of chosen terms is called a dictionary. The process of selecting the set of terms is called dictionary construction. Second, specific terms from the dictionary must be selected for use in describing each document. This assignment of terms to describe documents is called context analysis or document indexing. Both dictionary construction and document indexing have been previously investigated, with both manual and automatic methods considered. A large degree of success has been found in using fully automatic procedures for document indexing. [3,4,5] Automatic dictionary construction has not proved so successful and it is this area which is being presently investigated.

Dictionary construction may be conveniently considered in terms of several specific areas.

(i) NEGATIVE DICTIONARY CONSTRUCTION.

The determination of which terms to exclude from the document indexing process. This is defined more explicitly in the next section.

(ii) WORD STEMMING.

Entries in a dictionary may be grouped according to stems by means of suffixing. This involves construction of both Word Form and Word Stem dictionaries.

(iii) THESAURUS CONSTRUCTION.

Terms having similar properties may be clustered to form a single dictionary entry. These may be hierarchical in structure and may be based on many different similarity properties.

(iv) PHRASE DICTIONARIES.

Words or concepts used frequently in combination are identified.

(v) DICTIONARY UPDATING.

The dictionary for a dynamic document collection must also be dynamic. This involves updating of the dictionary as documents are added to or deleted from the collection, or as word usage changes.

The remainder of this paper deals with the first of these areas; negative dictionary construction. Further background of this specific area is given in the following section.

2. Negative Dictionaries

2.1 Common Words

The words used in the text of a document may be divided into two classes, which might be described as "words important to the *meaning* of the document" and "words important only to the *structure* of the document". For example, consider the phrase;

> "The role of the generality effort in retrieval system evaluation is assessed, ..."

as used in a document in the field of information retrieval. Intuitively, the words in this phrase could be divided into the following two classes:

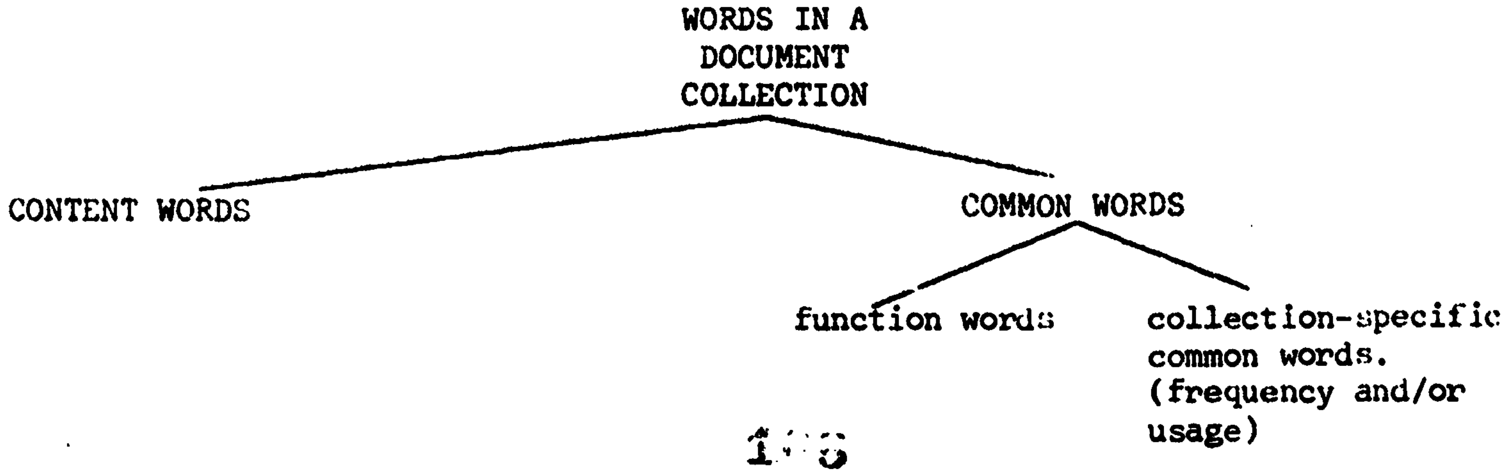
"MEANING" WORDS	"STRUCTURE" WORDS
ROLE	THE
GENERALITY	OF
EFFECT	IN
RETRIEVAL	IS
SYSTEM	
EVALUATION	
ASSESSED	

Those words which are important only to the structure of the documents are called *function* words or *common* words. Those words which are important to the meaning of the sentence are called *content* words. If only function words are classified as common, then the process of negative dictionary construction is not difficult. However, a closer examination of the previous example reveals some further problems, indicating that the class of common words should possibly be expanded to include words other than function words.

Consider the use of the word "retrieval" in the above example. Since this was taken from a document in the field of information retrieval, some question may be raised as to the validity of using the word "retrieval" to describe any document. It might be expected that many, or even all of the documents in this collection would contain this word. Although using "retrieval" to index each document in which it occurs may contribute to the accuracy of the description of those documents, it will also serve to make distinguishing among those documents more difficult. For this reason, words which have a very high frequency of occurrence in a collection may be considered to be common words.

Again examining the example given, consider the use of the word "role". Although not strictly a function word, "role" would not appear crucial to the meaning of the phrase to the same extent as "generality" or "evaluation". In fact, the phrase could easily be reworded in several ways to eliminate completely the word "role". Thus in this collection, retrieval may be expedited by treating a word such as "role" as common because of its usage. In a collection dealing with the theatre, for example, "role" might in fact be a very important and meaningful word.

Words which are classed as common because of their high frequency of occurrence in a particular collection, or because of their specific usage in the collection are called collection-specific common words. Thus, it is convenient to classify words in the following manner:


WORDS IN A DOCUMENT COLLECTION
CONTENT WORDS
COMMON WORDS
function words
collection-specific common words. (frequency and/or usage)

A classification such as this may be useful in constructing a negative dictionary using manual methods. Manual construction of negative dictionaries is discussed further in Section 4. This model is not, however, as useful when automatic negative dictionary construction methods are considered. Thus, new approaches to the problem of classifying words in a collection have been considered, with the hope that classifications which may be stated in more precise mathematical terms may also be simpler to implement using automatic techniques. This notion is expanded in Section 5.

2.2 The Negative Dictionary

A negative dictionary is defined as a list of words whose use is proscribed for content analysis purposes. Based on the classification outlined in the previous section, the negative dictionary for a collection is composed of those terms in the collection which are either function words or collection-specific common words.

The importance of accurate construction of negative dictionaries has been demonstrated. Words which do not contribute to the effectiveness of the information retrieval process must be excluded. Bergmark [6] has shown that, in at least one case, the advantage of a thesaurus over a word stem dictionary was due to more accurate determination of common words, rather than to the clustering of the terms.

Negative dictionary construction is considered in detail in the following sections. Section 3 describes briefly the experimental procedures used, including the retrieval system, document collections, and evaluation methods. In Section 4 manual negative dictionary construction is described and some retrieval results presented. Section 5 includes discussion of possible techniques to be used in automatic construction of negative dictionaries. In Section 6, the techniques of Section 5 are incorporated into specific algorithms for use in negative dictionary construction. Several algorithms are tested for retrieval effectiveness and these results are presented. Finally the work is evaluated and conclusions are drawn in Section 7.

3. Experimental Procedures

The retrieval system, document collections, and evaluation parameters used in the experiments to follow are described briefly.

3.1 The SMART System

The SMART system is an automatic document retrieval system "designed for the exploration, testing, and measurement of proposed algorithms for document retrieval". [7] All the experiments discussed in the following sections were performed using the SMART system as the experimental base.

As described by Williamson, [7] the SMART system takes documents and search requests in natural language, performs a fully-automatic content analysis of the texts, matches analyzed documents with analyzed search requests, and retrieves those stored items believed to be most similar to the queries. A description of the implementation of the SMART system may be found in [7]. Reports of previous work done using the SMART system are numerous, including [8] and [9].

3.2 The Experimental Data Base

Two document collections are chosen for use in the retrieval experiments. The first is the Medlars collection of 1033 abstracts from the field of medicine, along with 35 queries for which relevancy judgements were obtained. The second is the ophthalmology collection consisting of 852 documents and 35 queries. Again, a set of relevancy decisions for each of the queries with each document was obtained. These collections are suitably large to yield valid results, yet are of a size which allows extensive experiments to be performed using the computer resources available.

3.3 Evaluation Parameters

Two principal measures have been chosen for use in evaluating the retrieval effectiveness of the methods being tested. [10] These measures are precision (P) and recall (R), which are defined as follows:

$$P = \frac{\text{number of relevant documents retrieved}}{\text{number of documents retrieved}}$$

$$R = \frac{\text{number of relevant documents retrieved}}{\text{number of relevant documents in the collection}}$$

Thus, precision is the percentage of retrieved documents which are actually relevant, whereas recall is the percentage of relevant documents actually retrieved. In presenting retrieval results, these recall and precision values are averaged over all search requests and displayed in the form of a graph. The performance of different methods is compared using these precision-recall graphs.

4. Manual Negative Dictionary Construction

4.1 Methods For Manual Negative Dictionary Construction

Dictionaries and keyword lists used for content analysis purposes always include a negative dictionary. Thus, the dictionary construction process involves partitioning ehe terms in a collection into two sets of terms; those terms which are to be included in the indexing of the documents (the inclusion list), and those terms to be excluded from the indexing (the exclusion list). Generally, manual construction of a dictionary proceeds from either of two directions. In one case, the keyword list or inclusion list is selected from all the words in the collection. The remaining terms thus form the exclusion list or negative dictionary. In the other case, the emphasis is placed on determining which terms to exclude from the indexing process and it is the negative dictionary which is constructed first. Thus the remaining terms in this case form the inclusion list. Although differing in description, these two approaches to dictionary construction are not too diverse. In each case, all distinct words in a collection are manually examined and a decision is made with regard to their usefulness in indexing the documents.

There are interesting examples of experimental work involving dictionary construction using each of the above approaches. In work performed by Vaswani and Cameron, [11] a dictiona..y was constructed from a sample of 1,648 abstracts. Initially, a list was constructed showing each word occurring in the sample, along with the number of times the word occurred. The method of then constructing the dictionary proceeded as follows:

> "The list was studied very carefully by three people, two of them being fairly familiar with the subject matter, who decided intuitively which words to retain in the system as keywords, all others being excluded from further consideration".

Thus, the negative dictionary consisted of those terms "excluded from further consideration".

In document retrieval experiments done using the SMART system, the following procedure has been used for constructing negative dictionaries: [12]

(i) A standard common word list is prepared consisting of function words to be excluded from the dictionary;

(ii) A concordance listing is generated for a sample of the document collection under consideration, giving the context and the total frequency of occurrence for each word;

(iii) The common word list is extended by adding new non-significant words taken from the concordance listing; many of the words added to form the negative dictionary are either very high frequency words providing little discrimination in the subject area under consideration, or very low frequency words which produce few matches between queries and documents.

The use of automatically generated aids such as concordance listings and word frequency counts has proved helpful during manual dictionary construction. Nevertheless, the construction process still involves an intellectual decision with regard to each term in the collection, and many of these decisions must still be made somewhat intuitively.

4.2 Manual Negative Dictionary Construction-Performance Results

Using the manual negative dictionary construction method outlined in the previous section, a negative dictionary was constructed for the Medlars collection. The remaining (non-excluded) terms were then processed three separate ways to produce the following three dictionaries:

(i) The M1-word form (suffix-'s') dictionary, formed by stripping the final s from all terms;

(ii) The M2-word stem dictionary, formed by automatic removal of suffixes as determined from a previously prepared standard suffix list;

(iii) The M3-thesaurus, or synonym dictionary, formed by manually grouping dictionary entries into synonym categories, or concept classes.

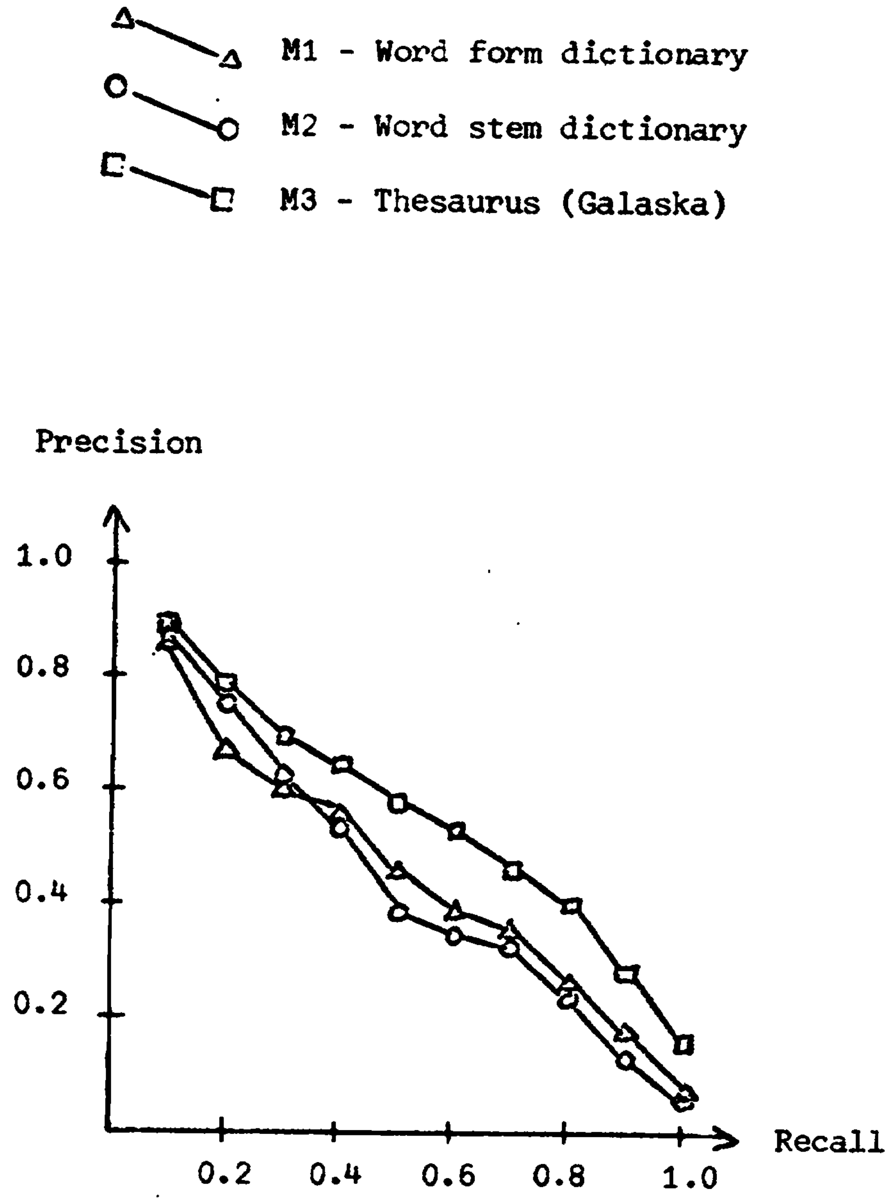
A recall-precision graph showing the performance of these three dictionary types is given in Fig. 1.

The performance of the M3 thesaurus is clearly better than that of either the M1 or M2 dictionaries, and may be attributed to a combination of accurate common word recognition and careful term clustering. The performance of the M1 word form and the M2 word stem dictionaries is quite similar; however, the M1 dictionary gives better performance results at all recall points except in the range of .10 to .30.

Based on these results, consideration is given as to which dictionary type to use for testing of automatic negative dictionary construction methods. The simple word form dictionary type is selected for several reasons. First of all, use of a thesaurus presents both construction and analysis problems; it may be difficult to determine whether performance changes are due to common word recognition or to term clustering. Secondly, the performance of the word stem dictionary in the manual case gives no reason to select it over the word form dictionary. Finally, the word form dictionary is the simplest to construct. Therefore, the M1 word form dictionary is selected as the "control" dictionary for use in comparing the effectiveness of manual and automatic negative dictionary construction methods.

5. Automatic Methods of Negative Dictionary Construction

In approaching the problem of automatic negative dictionary construction, a first course of action may be to adhere closely to one of the algorithms used in manual negative dictionary construction, attempting to



R	Precision		
	△——△	○——○	□——□
0.1	.8253	.8381	.8384
0.3	.5985	.6198	.6804
0.5	.4304	.3990	.5751
0.7	.3252	.3111	.4509
0.9	.1703	.1338	.2765

Dictionaries Constructed Using Manual
Deletion of Common Words
Fig. 1

automate each step of the manual process. Examples of the difficulties found when this approach is used may be seen in each of the manual methods outlined in the previous section. In the case of the method used by Vaswani and Cameron, the problem arises at the point at which the manual worker "decides intuitively" which terms to include in the dictionary. It is difficult to conceive an automatic procedure which will duplicate the intuitive decisions made by an individual. In the case of the manual negative dictionary construction procedure used on the SMART system, the problem of automation arises in the step involving examination of the concordance listing. Although a manual worker may with high consistency locate collection-specific common words by examining a concordance listing, this process does not yield directly to automatic methods. Based on these considerations, it is worthwhile to develop another approach to the problem of automating the negative dictionary construction process.

The approach that is followed is to consider factors regarding terms in a collection which may be measured and evaluated objectively. Several factors are considered and tested. These are divided into the following three areas:

(i) frequency and distribution,

(ii) discrimination value,

(iii) distribution correlation.

In the following three sections, each of these areas is discussed in detail.

5.1 Frequency and Distribution of Terms

Three basic statistics are considered for use in determining which terms belong in the negative dictionary for a collection. These values, which may be easily computed for each term in a collection, are: *total frequency*, which is the total number of occurrences of the term in the collection; *document frequency*, which is the number of documents in the collection in which the term occurs at least once; and *average usage*, which gives the average number of times the term is used within the documents in which it actually occurs (this is simply total frequency divided by document frequency). For discussion purposes, three separate areas are considered, in which these statistics are utilized. Terms of low frequency are discussed in Section 5.1.1, terms of high frequency are discussed in Section 5.1.2, and a discussion of the average usage of terms is given in Section 5.1.3.

5.1.1 Low Frequency

A consideration of terms with a low frequency of occurrence is important due to the fact that the *majority* of the terms in a collection occur only a very few times. For example, in the Medlars collection of 1033 medical abstracts, there are 14,534 unique terms in the text. Table 1 lists the number and percentage of terms with specific low total frequencies in this collection. It can be seen from this table that words of total frequency *one* account for almost half the unique occurrences in the collection.

Consideration is given to placing this large number of single occurrence terms on the exclusion list, and two advantages of doing so are noted. First of all, many of these terms are actually errors in the text, such as misspellings, improper hyphenation, etc. Excluding these terms causes elimination (but not correction) of these errors during the document

TOTAL FREQUENCY	NUMBER OF TERMS	PERCENTAGE OF TERMS	TOTAL OCCURRENCES
1	7,065	48%	7,065
2	2,073	15%	4,146
3	937	6%	2,811
over 3	4,459	31%	146,526
TOTAL	14,534	100%	160,548

Number of Low Frequency Terms

(Medlars Collection)

Table 1

indexing process. Second of all, the size of the inclusion list is kept much smaller by excluding single frequency terms. Although this is an efficiency consideration, which may be considered as of lesser importance than retrieval effectiveness, maintaining a dictionary of a size which may be handled is an important factor.

Regardless of these two advantages, it is the effect of terms of single occurrence on retrieval effectiveness which must be considered.

Very low frequency words may be expected to produce few matches between queries and documents. [13] It may be argued that the few matches which do occur will be important, and that low frequency words should therefore be retained. Excluding very low frequency terms may therefore cause a decrease in the level of precision of the retrieval results for some queries.

For the two document collections used in this study, all terms occurring once in a collection were matched against the terms in the queries for that collection. In no case was there a match. All terms having a total frequency of one could therefore be excluded without any resulting loss in precision of retrieval results. Because it is difficult to generalize these results to either the case of more queries in these present collections, or to the case of larger collections, it is clear that some compromise must be made regarding low frequency terms. This compromise is between retrieval effectiveness and retrieval efficiency. When terms of very low total frequency are excluded, the dictionary is kept small, increasing efficiency, but the level of precision may drop, indicating a decrease in retrieval effectiveness. The choice of a particular value of total frequency for use in excluding low frequency terms depends on the levels of retrieval effectiveness and efficiency which are required.

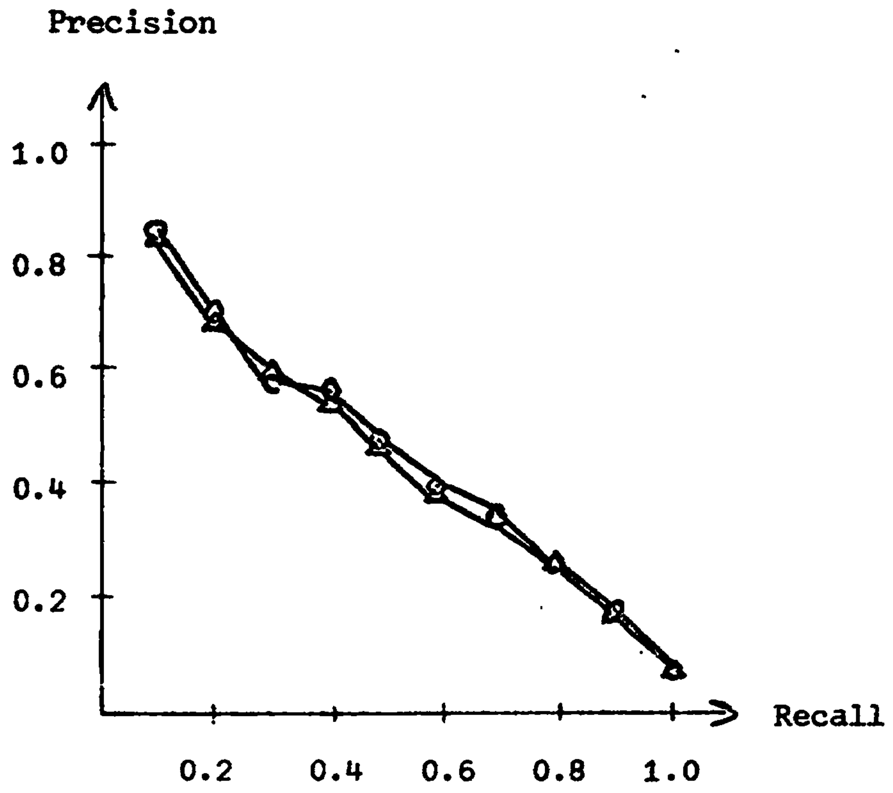
The effect of deleting terms of total frequency one from the Medlars collection is investigated experimentally. All 5,718 terms of frequency one are deleted from the M1 word form dictionary to form the MA word form (no frequency one) dictionary. The comparative performance of the M1 and MA dictionaries is shown by means of a precision-recall graph in Fig. 2. As discussed previously none of the deleted terms matched with any of the query terms, so the performance of the M1 and MA dictionaries should be similar. This is verified by the results shown in Fig. 2. Those slight changes which do occur are a result of the decrease in the lengths of the document vectors due to deletion of the low frequency terms.

5.1.2 High Frequency

Very high frequency words provide little or no discrimination in the subject area of the document collection under consideration. It may, therefore, be worthwhile for very high frequency words to be placed on the exclusion list. Excluding very high frequency terms from the indexing process may result in some decrease in the level of recall for certain queries. [13] That is, there may be some documents relevant to a particular query which are not retrieved by that query due to the exclusion of one or more high frequency terms which would have provided a match between the query and the documents. However, if high frequency terms are not excluded from the indexing process, then a query may match significantly with documents which are quite dissimilar. This may result in a low level of precision for certain queries.

△——△ M1 - Word form dictionary

○——○ MA - Word form dictionary - terms of frequency one deleted



R	Precision	
	△——△	○——○
0.1	.8253	.8276
0.3	.5985	.5885
0.5	.4304	.4396
0.7	.3252	.3253
0.9	.1703	.1721

Manually Constructed Word Form Dictionaries

Effect of Frequency One Terms

Fig. 2

Obviously, some compromise is necessary between the desire for high recall and the need for high precision. It may be possible, however, to delete very high frequency terms so that precision increases, without affecting recall to a very great extent. What is desired is a function, based on either total frequency, or document frequency, or both, which will enable determination of those high frequency terms which belong on the negative exclusion list.

Consider the list of words in Table 2. These are the terms of highest frequency in the Medlars collection of 1033 medical abstracts. The words are given in decreasing order of document frequency. By examining the total frequency given for each word, it is apparent that an ordering by total frequency would have been quite different. In particular, words such as CELL and CASE would occur much higher on this list if it were ordered by total frequency.

The three lines drawn through Table 2 indicate levels of document frequency of 20, 25, and 30 percent of total collection size. All the terms occurring with a document frequency of 30% of collection size or greater are clearly function words and belong in the negative dictonary for this collection. On the other hand, above a document frequency level of 20% of collection size there are several terms which are clearly not function words. At the document frequency level of 25% of collection size, only the word PATIENT is not a function word, and it is clear that the word PATIENT could easily be a collection-specific common word in a medical collection.

Similar results to the above have been found for a collection of 852 abstracts in the field of ophthalmology, in that a document frequency

TERM	DOCUMENT FREQUENCY	TOTAL FREQUENCY
OF	1,027	9,327
THE	1,021	11,174
AND	990	4,799
IN	988	5,337
A	868	2,681
TO	856	2,673
WITH	759	1,884
IS	637	1,580
BY	600	1,203
WAS	561	1,492
THAT	537	970
FOR	511	924
WERE	495	1,214
BE	480	821
FROM	444	831
AS	442	814
THIS	439	654
ON	436	744
ARE	431	750
IT	417	688
AN	412	637
NOT	405	606
OR	393	653
THESE	343	461
WHICH	328	494
AT	303	494
PATIENT	302	799
BEEN	268	388
BUT	268	348
AFTER	258	435
HAVE	256	332
CASE	253	521
OTHER	234	304
THAN	233	307
RESULT	227	285
HAS	225	305
MAY	222	327
FOUND	210	288
CELL	208	785
EFFECT	207	346
NORMAL	203	369

Medlars High Frequency Terms

Ordered by Document Frequency

Table 2

level of 25% of collection size provided a point above which terms could be placed on the exclusion list with a high degree of confidence.

Thus, terms with a document frequency greater than some chosen cutoff value should be placed in the negative dictionary. By choosing this cutoff value properly, precision may be improved without any significant reduction in recall.

5.1.3 Average Usage

It is worthwhile to investigate whether terms which belong in the negative dictionary may be distinguished by their average usage from terms which do not belong in the negative dictionary. In particular, it would seem reasonable that terms which are of importance in a collection may be used several times within the documents in which they occur, thus having a high average usage. On the other hand, function words might be expected to have a more random distribution, thus having a low or medium average usage. Average usage values were examined for over 6000 terms from a collection of 852 documents in the area of ophthalmology.

Table 3 lists some of the terms from this collection, ordered by average usage. An examination of this list shows that content words, such as LASER and CYST, cannot be distinguished from common words, such as WAS and TO by means of average usage values. Average usage is therefore rejected for use in automatic negative dictionary construction.

5.2 Discrimination Value

A document collection may be defined as a distribution of terms taken from a specified set of terms. Thus, each term may be considered as a possible index term on the basis of its distribution in relation to the distribution of all other terms in the collection. Developing this

TERM	AVERAGE USAGE
SPRAY	12.50
THE	9.38
OF	6.78
RUBELLA	4.70
CHOLINESTERASE	4.33
IN	4.15
AND	3.50
COLLAGEN	3.33
FIBER	3.13
CYCLODIALYSIS	3.00
LASER	2.91
CAPILLARY	2.84
A	2.67
TO	2.50
CYST	2.47
WAS	2.39
VESSEL	2.28
UVEITIS	2.14
EYE	2.14
IS	2.13

Average Usage of Certain Terms
From Ophthalmology Abstracts
Table 3

idea, a term is considered to be a *discriminator* if its distribution is such that it serves in distinguishing or discriminating among the documents in the collection. A term which does not serve in distinguishing among the documents in the collection is a *non-discriminator*. For example, any term which occurs in all the documents in a collection is a non-discriminator in that collection, as it may not be used to distinguish among the documents in any way.

It is useful then, to define some function for terms in a collection which would indicate whether they are discriminators or non-discriminators. Such a function, the *discrimination value*, is suggested, based on the document space similarity described by Aste-Tonsmann and Bonwit. [14] Some notation and definitions are given and the discrimination value is derived in the following section.

5.2.1 The Discrimination Value Function

A collection of N documents is represented by a set of document vectors $\underline{d}_i$, $1 \leq i \leq N$. Each vector $\underline{d}_i$ is of length m, where m is the number of terms used in indexing the documents in the collection. Then d_{ij} is the number of occurrences of the i^{th} term in the j^{th} document. Therefore, a value of $d_{ij} = 0$ indicates that the i^{th} term did not occur in document j. The centroid, $\underline{c}$, of a document collection is defined by $\underline{c} = (c_1, c_2, \ldots, c_m)$ where:

$$c_i = \frac{1}{N} \sum_{j=1}^{N} d_{ij} \ .$$

The centroid represents a term by term average of the documents and is considered to be the center of the set of documents (i.e. of the *document space*).

A measure of the correlation of two vectors $\underline{d}_k$ and $\underline{d}_1$ is given by the cosine function:

$$\cos(\underline{d}_k, \underline{d}_\ell) = \frac{(\underline{d}_k, \underline{d}_1)}{||\underline{d}_k||\ ||\underline{d}_1||}$$

where $(\underline{d}_k, \underline{d}_1)$ is the inner product and $||\underline{d}_k||^2 = (\underline{d}_k, \underline{d}_k)$. For purposes of calculation, this is conveniently expressed as:

$$\cos(\underline{d}_k, \underline{d}_1) = \frac{\sum d_{ik} \cdot d_{i1}}{\sqrt{\sum d_{ik}^2 \sum d_{i1}^2}}$$

where the sums are for $i = 1$ to m, the number of terms in the vectors.

Now the compactness or document space similarity, Q, is defined as:

$$Q = \frac{1}{N} \sum_{j=1}^{N} \cos(\underline{c}, \underline{d}_i)\ ,\ 0 < Q \leq 1. \qquad \text{(i)}\ \text{(ii)}$$

The value of Q is thus a function of the homogeneity of the documents in the collection and the set of terms used in indexing the documents. Given a standard index language, a collection of documents from diverse subject areas will tend to have a low Q value, whereas a collection of documents on very similar topics will tend to have a higher Q value. However, for more homogeneous collections, the proper selection of index terms will reduce the compactness of the document vectors, enabling better discrimination and resulting in improved retrieval.

(i) This is the "NORMALIZED Q" of Aste-Tonsman and Bonwit. It is a more convenient measure than their Q and has a well defined range.

(ii) This may be maximized for a Rocchio centroid $\underline{c}' = \frac{1}{N} \sum \frac{d_i}{||d_i||}$ and not for $\underline{c}$ as defined. However, this is negligible for docs of approximately the same length.

Since Q is a function of the document vectors, deleting a single term from all of the document vectors will normally change the value of Q. Essentially, deletion of this term represents a new index language, differing from the initial one by only one term. The compactness of the collection with term i deleted (i.e. $d_{ij} = 0, 0 \leq j \leq N$) is given by Q_i and is defined as:

$$Q_i = \frac{1}{N} \sum_{j=1}^{N} \cos(\underline{c}^i, \underline{d}_j^i)$$

where $\underline{d}_j^i$ is the j^{th} document vector with term i deleted, and $\underline{c}^i$ is the centroid vector with term i deleted.

Then $(Q_i - Q)$ is a measure of the change in document space compactness due to the deletion of term i. If $Q_i > Q$, the document space is <u>more compact</u> with term i deleted and term i is a <u>discriminator</u> in the collection. If $Q_i < Q$, the document space is <u>less compact</u> with term i deleted and term i is a <u>non-discriminator</u> in the collection. Since a particular Q_i value is only meaningful in comparison to the value of Q, a new measure is defined. The <u>discrimination value</u> D_i of term i is defined as:

$$D_i = \frac{Q_i - Q}{Q} * 100$$

D_i has the following properties:

(a) $D_i \leq 0$, term i is a non-discriminator

(b) $D_i > 0$, term i is a discriminator

(c) $D_i < D_j$, term j is a better discriminator than term i.

(d) D_i is not an explicit function of collection size, allowing comparison of values of D_i computed for a term i occurring in several collections.

The discrimination value thus provides a function by which all terms in a collection may be ranked, from greatest non-discriminator to best discriminator.

It is suggested that for each document collection, a discrimination cutoff value exists such that all terms with a discrimination value below this cutoff value should be placed in the negative dictionary for the collection. Only terms with a discrimination value greater than the chosen cutoff value are used in indexing the documents. It is further suggested that the discrimination cutoff value for any collection will be strictly non-negative. That is, non-discriminators should always be placed in the negative dictionary for a collection. For a discrimination cutoff value, D_c, a term i in a collection may be classified as follows:

$D_i \leq 0$, term i is a non-discriminator

$0 < D_i \leq D_c$, term i is a <u>poor</u> discriminator

$D_c < D_i$, term i is a <u>good</u> discriminator.

The effective use of discrimination value in an algorithm for negative dictionary construction is demonstrated in Section 6. The determination of a discrimination cutoff value and its usefulness are also shown.

5.2.2 The Set of Non-Discriminators

Discussion in the previous section concerned the computation of discrimination value for specific terms in a collection. However, a collection of documents includes a large number of terms, many of which bear relation to one another. Thus, conclusions which may be drawn for individual terms do not necessarily hold true for groups of terms. For a given set of terms, each of which is a non-discriminator, it must be considered whether the set of terms also acts in a non-discriminary way.

Stated simply for two terms in a collection, the question is as follows. When the effect of deleting term i alone is known, and the effect of deleting term j alone is known, what conclusions may be drawn regarding the effect of deleting both terms i and j? For example, consider two terms i and j such that $D_i < 0$ and $D_j < 0$; is it true that $D_{ij} < 0$, where D_{ij} is defined as the discrimination value of the set of terms {i, j}. It can be shown that this is in fact true. That is, if terms i and j are non-discriminators, then they also act together in a non-discrimination way.

THEOREM 1 Let K and L be terms in a document collection such that $D_K < 0$ and $D_L < 0$. Then $D_{KL} < 0$.

PROOF The compactness of a collection with terms K and L deleted is defined as:

$$Q_{K,L} = \frac{1}{N} \sum_{i=1}^{N} \cos(\underline{c}^{K,L}, \underline{d}_i^{K,L}) \tag{5.1}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \frac{(\underline{c}, \underline{d}_i) - c_K d_{Ki} - c_L d_{Li}}{(||\underline{c}||^2 - (c_K^2 + c_L^2))\ 1/2\ (||\underline{d}_i||^2 - (d_{Ki}^2 + d_{Li}^2))\ 1/2}$$

The assumption may be made that the vectors are large compared to any one term. Therefore:

$$\frac{{c_K}^2 + {c_L}^2}{||\underline{c}||^2} < < 1 \quad \text{and} \quad \frac{{d_{Ki}}^2 + {d_{Li}}^2}{||\underline{d}_i||^2} < < 1 \tag{5.2}$$

Thus, expanding the denominators as a binomial series:

$$Q_{K,L} = \frac{1}{N} \sum_{i=1}^{N} \frac{(\underline{c},\ \underline{d}_i) - c_K d_{Ki} - c_i d_L i}{||\underline{c}||\ ||\underline{d}_i||} (1 + \frac{1}{2} \frac{c_K^{\ 2} + c_L^{\ 2}}{||\underline{c}||^2}) (1 + \frac{1}{2} \frac{d_{Ki}^{\ 2} + d_{Li}^{\ 2}}{||\underline{d}_i||^2}))$$

$$= \frac{1}{N} \sum_{i=1}^{N} \frac{(\underline{c},\ \underline{d}_i) - c_K d_{Ki} - c_i d_{Li}}{||\underline{c}||\ ||\underline{d}_i||} (1 + \frac{1}{2} \frac{c_K^{\ 2}}{||\underline{c}||^2} + \frac{c_L^{\ 2}}{||\underline{c}||^2} + \frac{d_{Ki}^{\ 2}}{||\underline{d}_i||^2}$$

$$+ \frac{d_{Li}^{\ 2}}{||\underline{d}_i||^2})) \tag{5.3}$$

by dropping the last term.

Let:

$$\beta_{Li} = \frac{1}{2} (\frac{c_K^{\ 2}}{||\underline{c}||^2} + \frac{d_K^{\ 2}}{||\underline{d}_i||^2}) \tag{5.4}$$

$$\beta_{Ki} = \frac{1}{2} (\frac{c_L^{\ 2}}{||\underline{c}||^2} + \frac{d_{Li}^{\ 2}}{||\underline{d}_i||^2}) \tag{5.5}$$

Then,

$$Q_{K,L} = \frac{1}{N} (\sum_{i=1}^{N} \frac{(\underline{c},\ \underline{d}_i)}{||\underline{c}||\ \ ||\underline{d}_i||} + \sum_{i=1}^{N} \frac{(\underline{c},\ \underline{d}_i)}{||\underline{c}||\ \ ||\underline{d}||} \beta_{Ki} + \sum_{i=1}^{N} \frac{(\underline{c},\ \underline{d}_i)}{||\underline{c}||\ \ ||\underline{d}_i||} \beta_{Li}$$

$$- \sum_{i=1}^{N} \frac{c_K d_{Ki}}{||\underline{c}||\ \ ||\underline{d}_i||} (1+\beta_{Ki}+\beta_{Li}) - \sum_{i=1}^{N} \frac{c_L d_{Li}}{||\underline{c}||\ \ ||\underline{d}_i||} (1 \cdot \beta_{Ki}+\beta_{Li}))$$

$$= Q + \frac{1}{N} (\sum_{i=1}^{N} \frac{(\underline{c},\underline{d}_i)}{||\underline{c}||\ \ ||\underline{d}_i||} \beta_{Ki} - \sum \frac{c_K d_{Ki}}{||\underline{c}||\ \ ||\underline{d}_i||} (1+\beta_{Ki})$$

$$- \sum_{i=1}^{N} \frac{c_K d_{Ki}}{||\underline{c}||\ \ ||\underline{d}_i||} \beta_{Li} + \sum_{i=1}^{N} \frac{c_L d_{Li}}{||\underline{c}||\ \ ||\underline{d}_i||} (1+\beta_{Li})$$

$$- \sum_{i=1}^{N} \frac{c_L d_{Li}}{||\underline{c}||\ \ ||\underline{d}_i||} \beta_{Ki} \tag{5.6}$$

$$Q_{K,L} = Q + (Q_K - Q) - \frac{1}{N} \sum_{i=1}^{N} \frac{c_K d_{Ki}}{||\underline{c}|| \ \ ||\underline{d}_i||} \beta_{Li} + (Q_L - Q)$$

$$- \frac{1}{N} \sum_{i=1}^{N} \frac{c_L d_{Li}}{||\underline{c}|| \ \ ||\underline{d}_i||} \beta_{Ki}$$

$$= Q + (Q_K - Q) + (Q_L - Q) - R_{K,L} \qquad (5.7)$$

where

$$R_{K,L} = \left(\frac{1}{N} \sum_{i=1}^{N} \frac{c_K d_{Ki}}{||\underline{c}|| \ \ ||\underline{d}_i||} \beta_{Li} + \frac{1}{N} \sum_{i=1}^{N} \frac{c_L d_{Li}}{||\underline{c}|| \ \ ||\underline{d}_i||} \beta_{Ki}\right)$$

Substracting Q from both sides and dividing by Q yields.

$$\frac{Q_{K,L} - Q}{Q} = \frac{(Q_K - Q)}{Q} + \frac{(Q_L - Q)}{Q} - \frac{R_{K,L}}{Q}$$

Therefore:

$$D_{K,L} = D_K + D_L - \frac{R_{K,L}}{Q} \qquad (5.8)$$

Since $R_{K,L}$ is strictly positive and $Q > 0$, then $D_{K,L} < 0$ Q.E.D.

Having shown that two terms which are non-discriminators also act together as a non-discriminating set in the collection, it is a simple extension to prove a similar result for all non-discriminators in a collection.

COROLLARY 1

Let the set of terms
$S = \{s_1, s_2, s_3, \ldots, s_t\}$ be such that
$D_i \leq 0$ for all $i \varepsilon S$ and that
$D_i < 0$ for at least one $i \varepsilon S$. Then
D_s, defined as the discrimination value for the set of terms, is negative; $D_s < 0$.

PROOF

Since all the equations used in theorem 1 apply for sets of terms as well as single terms, the corollary is easily proved by successive application of theorem 1.

Thus, it may be concluded that deleting the set of all non-discriminators in a collection has the effect of making the collection less compact.

Further properties of the discrimination value function are examined in the next section.

5.2.3 Analysis of the Discrimination Value Function

It is of interest to consider further the properties of the discrimination value function. In particular, it is important to determine whether the discrimination value provides any new information regarding terms in a collection, or if in fact the same information is obtainable from term frequencies and distributions. Thus, the relationships between the frequency, the distribution, and the discrimination value of a term are considered.

Two approaches are used in investigating this area. First, theoretical consideration is given to the effect of various frequencies and distributions on the discrimination value. Second, experimental results are presented, demonstrating the relationships which do exist between discrimination value and the frequency and distribution of terms.

Yu and Wong [15] investigated the compactness function, Q_i, upon deletion of terms of various frequencies and distributions. Although specific conclusions could not be made, some general results were given. These are as follows:

(i) Any term occurring in nearly all of the documents is a <u>non-discriminator</u>, irrespective of the number of occurrences within each document. (This is intuitive, however it is not trivial to show).

(ii) For a collection of N documents, a term occurring in N' of the documents with a constant frequency, may be classified as a non-discriminator if the following inequality is satisfied:

$$\frac{N}{N'} \geq \frac{\|c\|^2}{c_K^2} \qquad \text{(i.e. } \frac{N}{N'} \text{ is large)}$$

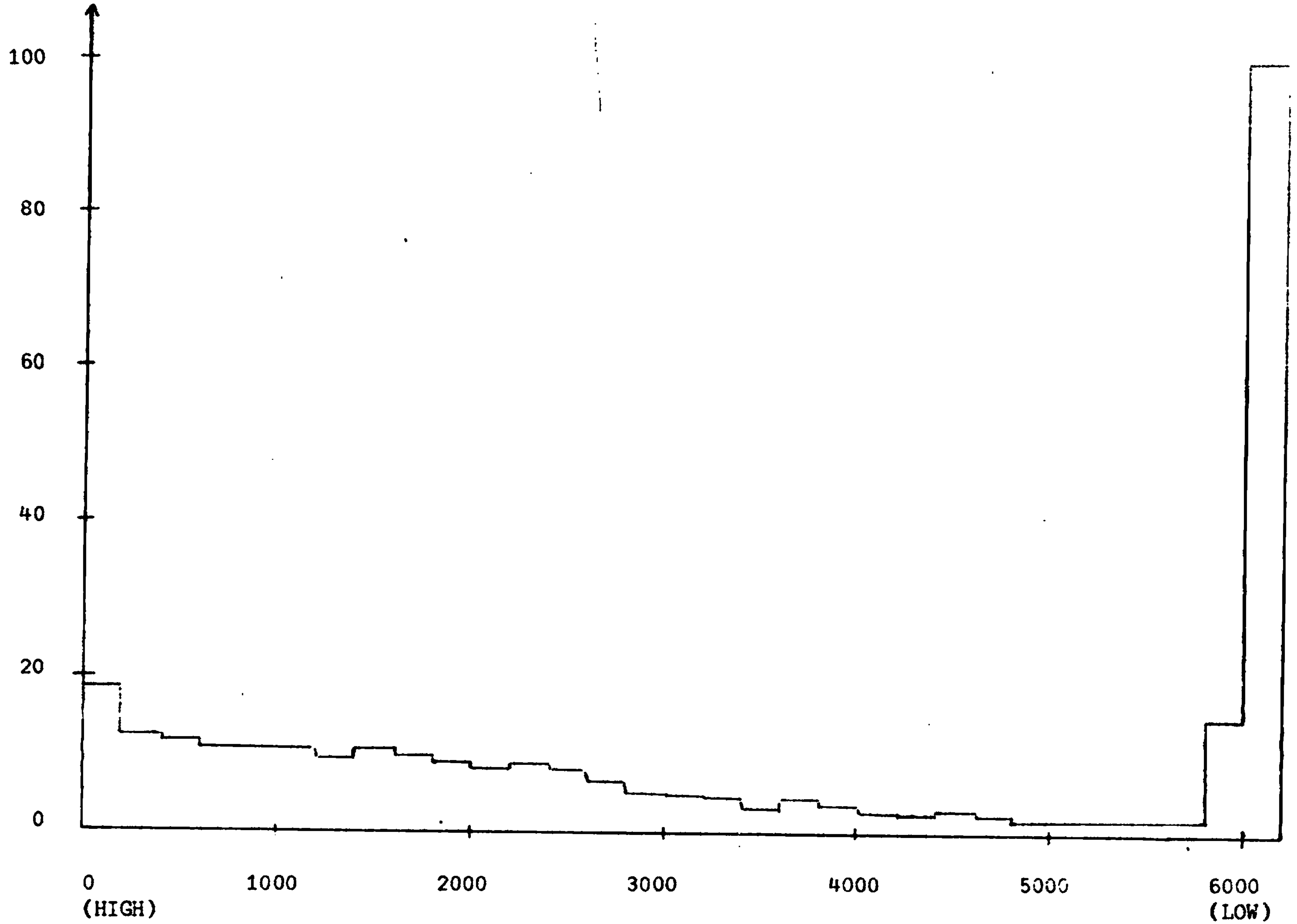
(iii) A term occurring with a bunched up distribution, in only a few documents, is a non-discriminator.

This third result (iii) may be invalid due to an assumption made by Yu and Wong that for a collection of N documents and m distinct terms,

$$\frac{N}{m^2} >> 1 \quad .$$

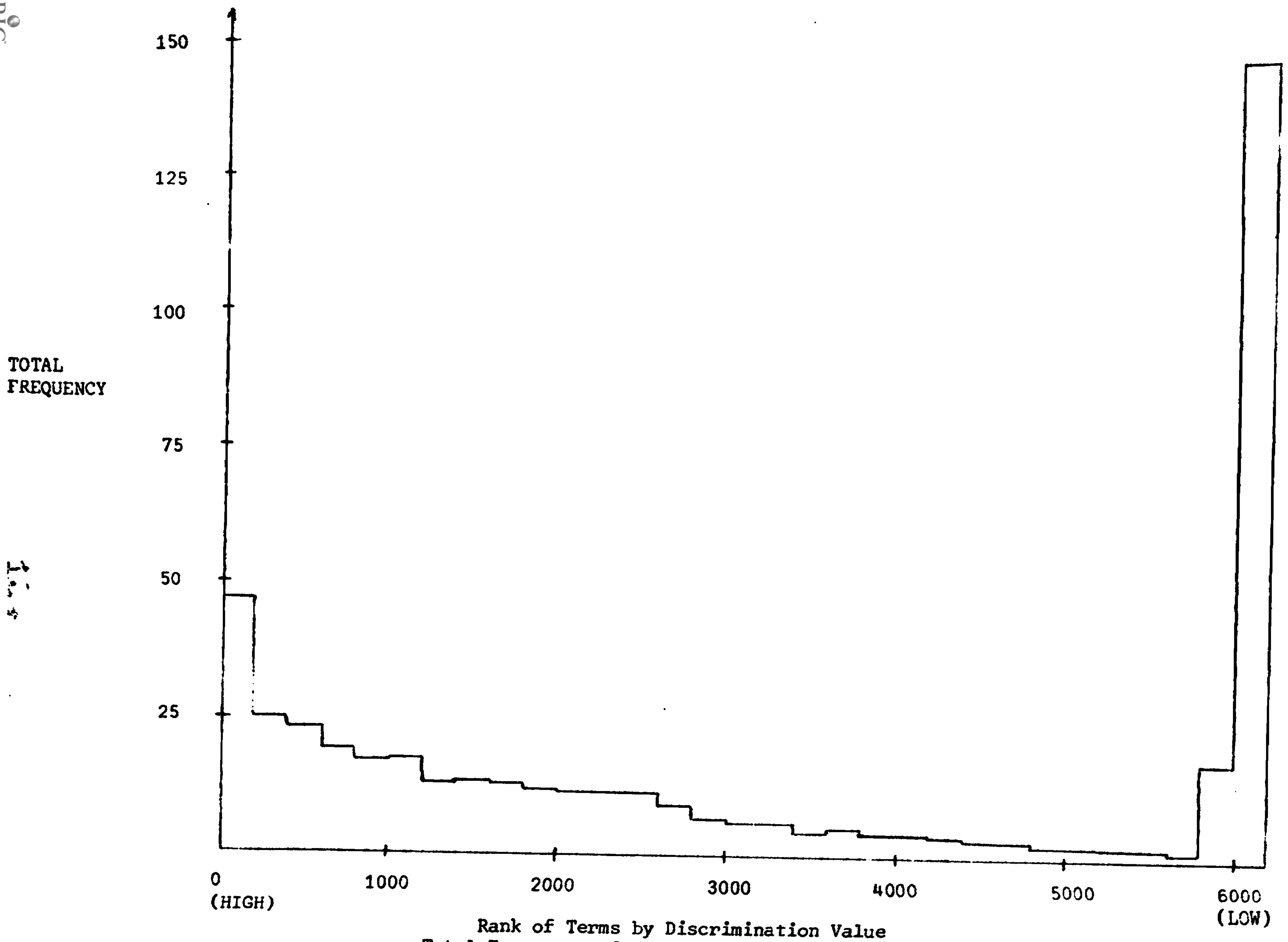
It is doubtful if this condition is met by any existing collections.

The Medlars collection was used to investigate experimentally the relationship between term frequencies and discrimination value. The discrimination value is computed for 6200 terms from this collection and these terms are then ordered by discrimination value. This ordered list is divided into 31 groups of 200 terms each. Thus the first group consists of the 200 terms with the highest discrimination values, and the last group contains the 200 terms with the lowest discrimination values. Averages are then computed giving the total frequency, document frequency, and average usage of each group of 200 terms. Figs. 3, 4, and 5 show these averages plotted in terms of the ordering by discrimination value.



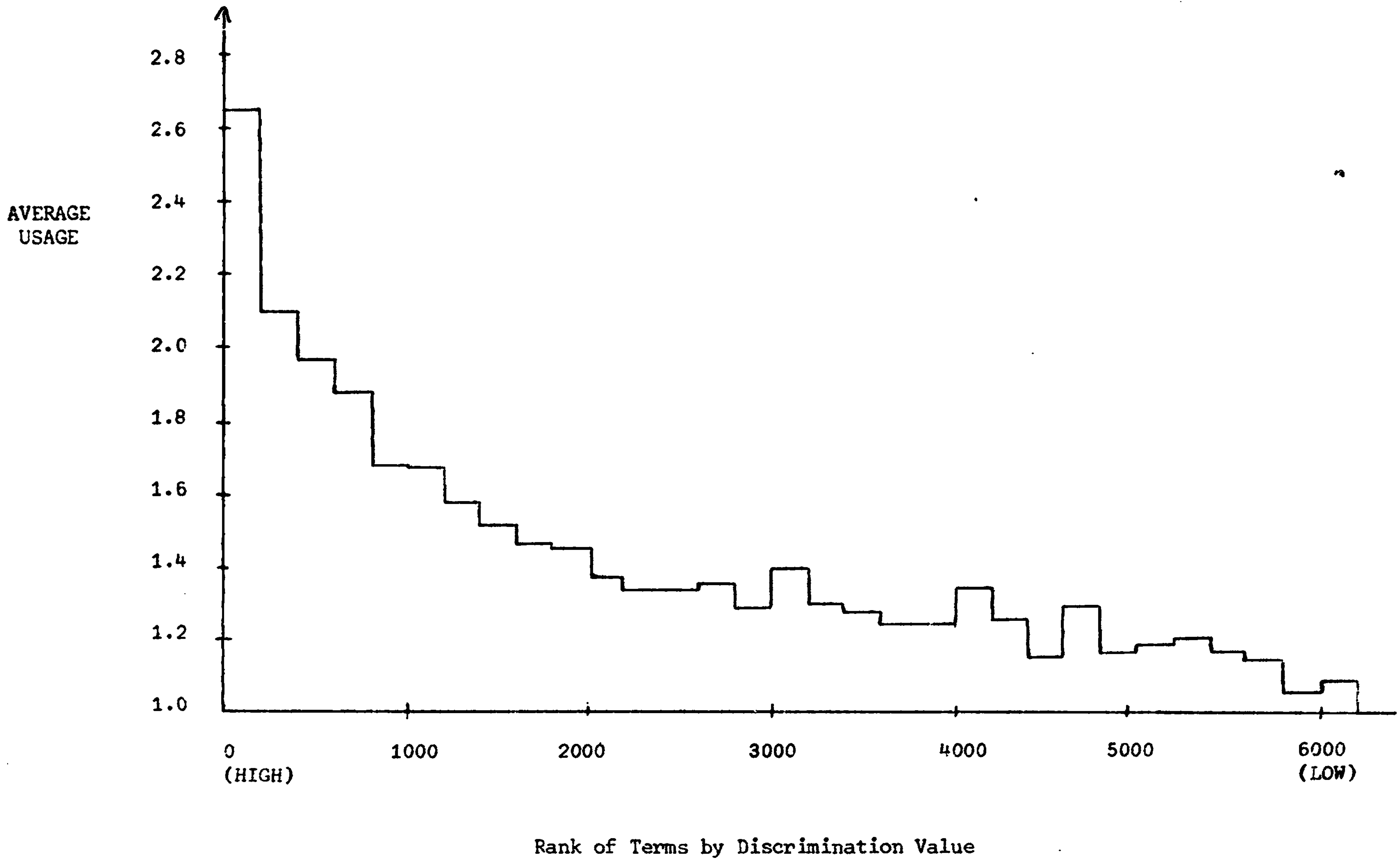
Rank of Terms by Discrimination Value
Document Frequency of Terms by Discrimination Value Rank

Fig. 3



Total Frequency of Terms by Discrimination Value Rank

Fig. 4



Average Usage of Terms by Discrimination Value Rank

Fig. 5

Consider Fig. 3 for example. The left end of the graph shows that the first (0-200) group of 200 terms (those with highest discrimination value) had an average document frequency of 18. That is, the best discriminators each occur in about 18 of the 1033 documents in the collection. Proceeding then from left to right across the graph, the figures indicate that as discrimination value decreases, so does document frequency, until those terms with the very lowest discrimination value are reached. At this point (5800), a large jump in document frequency occurs, with the group of 200 terms of lowest discrimination value having an average document frequency of about 100. The greatest difficulty in correlating discrimination value with document frequency arises with the next to last group of 200 terms of low discrimination value (5800-6000). In this case the average document frequency of the terms is almost 15, a very similar value to that of the best discriminators. Thus, very poor and very good discriminators cannot be distinguished by means of document frequency.

The results for total frequency given in Fig. 4 show a similar situation to the one just discussed. Again there are terms at either end of the discrimination value ordering which have similar total frequencies.

Fig. 5, which shows the relationship between discrimination value and average usage, gives indication that this relationship may be fairly direct. The group of terms with highest discrimination values (0-200) shows an average usage of over 2.6, that is, when these good discriminators are used within a document, they are used several times. As the discrimination value decreases, the average usage of terms also tends to decrease, although not monotonically. The results shown in Fig. 5 indicate that the average usage of a term may possibly be used to determine whether the term is a good discriminator or a non-discriminator. Since these results are averages, taken for groups of 200 terms, it is necessary to examine specific terms within the extreme groups (those with highest and lowest discrimination values). Table 4 presents the average usage values for 16 good discriminators and 16 non-discriminators, along with their document and total frequency values. These sample values are sufficient to show that discrimination value is apparently based on more than the frequency and distribution of a term.

Both the theoretical and experimental analysis of discrimination value, and its relationship to term frequency and distribution lead to the same conclusion. The discrimination value of a term does not depend only on the frequency and distribution of that term, but it depends also on the frequencies and distributions of all other terms in the collection.

5.3 Relative Distribution Correlation

A measure is presented for possible use in determining objectively which terms to place in the negative dictionary. It is hypothesized that content words and common words may be distinguished by the distribution of the documents in which they occur. A common word tends to occur in a somewhat random fashion, and the documents in which it occurs are not expected to bear any relation to each other in subject matter. A content word, on the other hand, tends to occur in documents of fairly homogeneous subject matter. For example, the words JUST and PANCREAS each occur 10 times in the Medlars documents. However, the documents in which they occur are distributed in the document space in a way exhibited in Fig. 6. The

NON-DISCRIMINATORS					DISCRIMINATORS				
TERM	DISCRIMINATION VALUE (X 100)	DOC. FRQ.	TOT. FRQ.	AVG. USAGE	TERM	DISCRIMINATION VALUE (X 100)	DOC. FRQ.	TOT. FRQ.	AVG. USAGE
PULMONARY	-.674	57	125	2.19	FRACTION	2.61	43	120	2.80
CHILD	-.532	52	118	2.27	HYPOTHERMIA	2.44	34	88	2.59
PROBLEM	-.463	49	73	1.49	MOTHER	2.43	34	87	2.56
COMPLETE	-.338	45	54	1.20	CENT	2.40	37	72	1.95
MAN	-.317	44	60	1.36	CULTURE	2.18	61	130	2.13
LOS (S)	-.295	46	61	1.33	LEN (S)	2.17	41	131	3.20
STILL	-.281	43	44	1.02	FLUID	2.13	44	94	2.14
RATHER	-.215	43	45	1.05	SYNTHESI (S)	2.02	42	78	1.86
APPEARANCE	-.164	41	46	1.12	DOG	1.91	43	106	2.47
RELATIVELY	-.145	41	44	1.07	URINARY	1.82	49	91	1.86
RESULTED	-.136	41	43	1.05	DAMAGE	1.80	34	53	1.56
EXAMINED	-.108	44	55	1.25	TEMPERATURE	1.66	36	73	2.03
KNOWN	-.108	41	45	1.10	DEATH	1.44	35	54	1.54
EXPERIMENTAL	-.102	44	53	1.20	EXCRETION	1.26	47	100	2.13
DISORDER	-.070	45	66	1.47	HEART	1.23	46	83	1.80
ORDER	-.057	34	36	1.06	STRUCTURE	1.21	41	68	1.65

Comparison of Frequencies for High Ranking
Discriminators and Non-Discriminators
(Medlars Collection of 1033 Abstracts)

Table 4

documents in which PANCREAS occurs are tightly grouped whereas the documents in which JUST occurs are spread throughout the document space.

A relative distribution correlation is suggested which indicates whether the documents in which a term occurs bear a strong or weak relationship. A term centroid, $\underline{c}_i$, is defined as the centroid vector for those documents in which term i occurs.

$$\underline{c}_i = \sum_{\substack{j \\ d_{ij} \neq 0}} \underline{d}_j$$

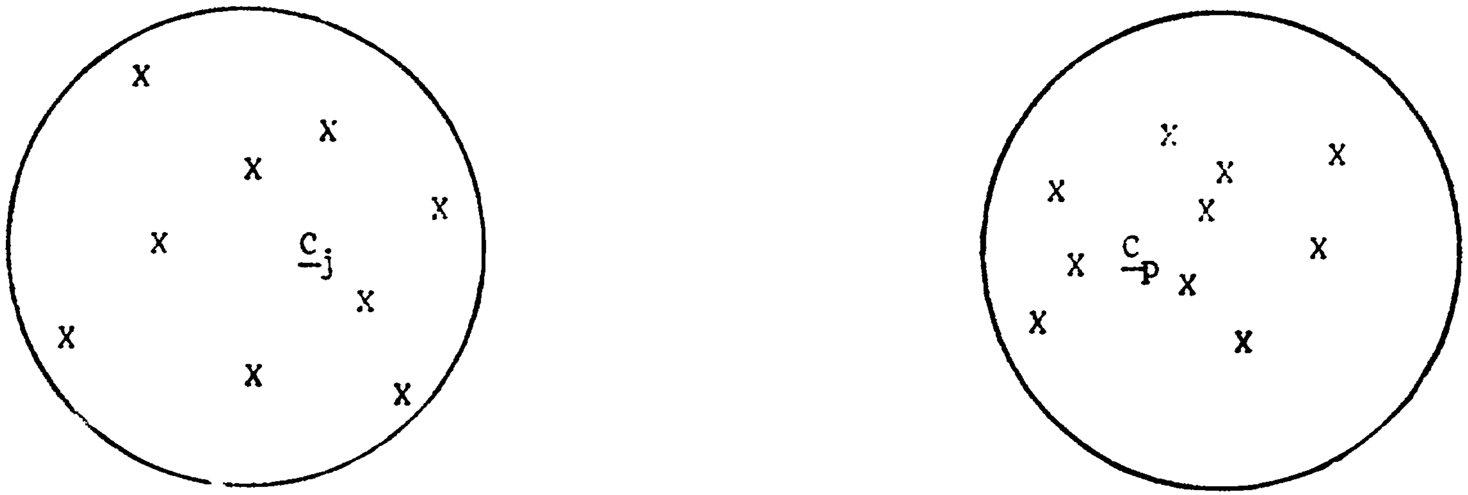
For example, Fig. 6 shows the term centroids $\underline{c}_j$ and $\underline{c}_p$ for the terms JUST and PANCREAS respectively. Then, assuming that term i occurs in n documents, the relative distribution correlation for term i, R_i, is defined as:

$$R_i = \frac{1}{n} \sum_{\substack{j \\ d_{ij} \neq 0}} \cos(\underline{c}_i, \underline{d}_j)$$

So that,

$$0 < R_i \leq 1 .$$

If the documents containing a term i are similar in content, then each of these documents correlates quite highly with the term centroid, $\underline{c}_i$, resulting in a value of R_i which is close to 1. Conversely, if the documents in which term i occurs are quite dissimilar, the value of R_i is close to 0. Consider again the example given in Fig. 6. The distribution of the documents results in a low relative distribution correlation for the word



Distribution in the document space of documents containing the word JUST. (C_j = centroid of the documents in which the term JUST occurs).

Distribution in the document space of documents containing the word PANCREAS. (C_p = centroid of the documents in which the term PANCREAS occurs).

Distribution of Common and Content Words in

The Document Space

Fig. 6

JUST but a somewhat higher correlation for the word PANCREAS.

The use of relative distribution correlation is investigated experimentally using a set of 20 terms. These terms, along with their relative distribution correlations are given in Table 5, ranked in order of correlation.

Several of these terms, such as ANTIGEN (rank 4) and ANESTHESIA (rank 11), are clearly content words. Other terms, such as WHOM (rank 6) and THOUGH (rank 16), are apparently common words. Yet this ranking of these terms by relative distribution correlation fails.to distinguish between common and content words. It is therefore concluded that the relative distribution relation does not provide information which is useful for purposes of negative dictionary construction.

6. Experiments and Results

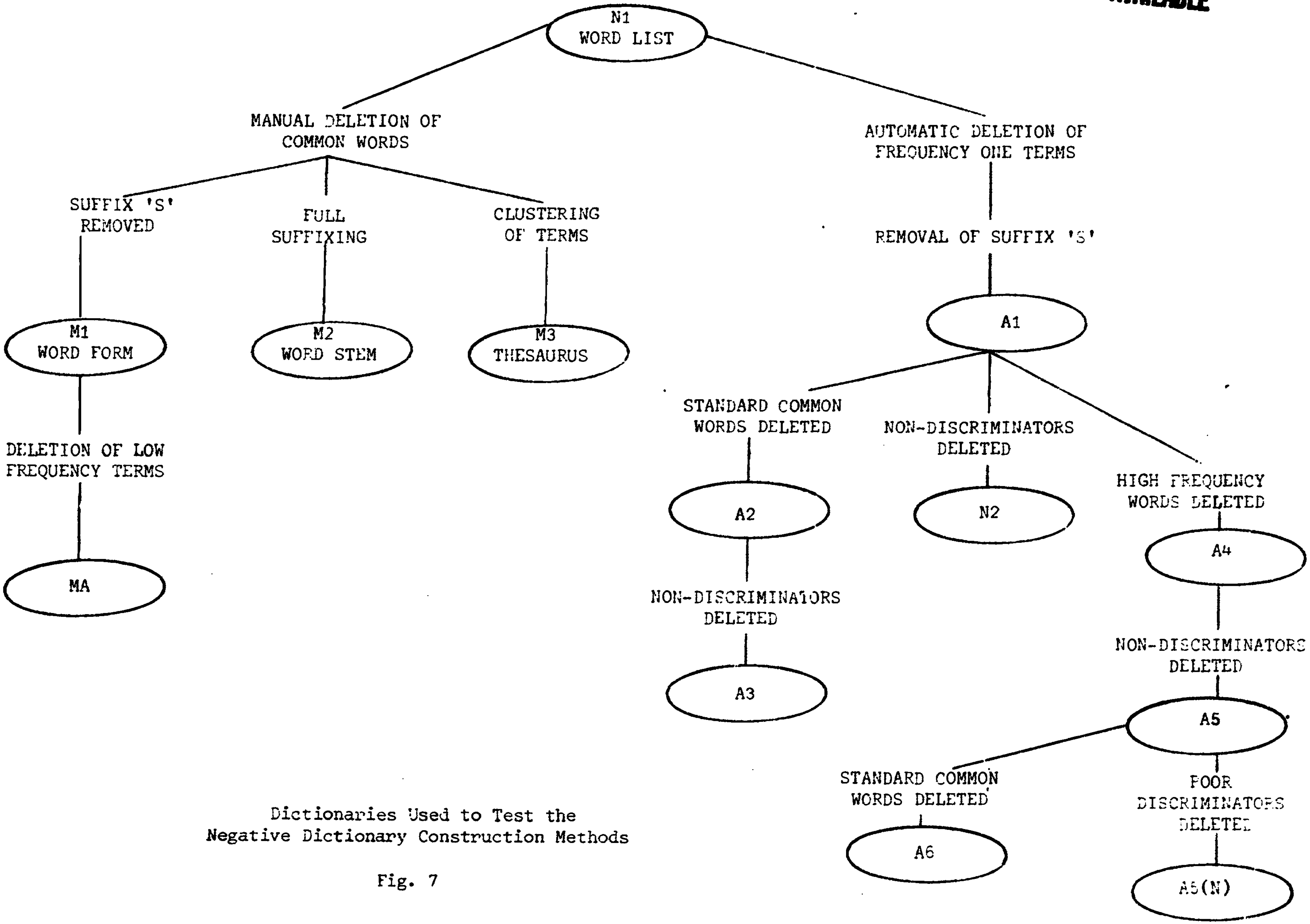
An experimental procedure for testing of the negative dictionary construction methods already discussed is outlined in Fig. 7. Each node in Fig. 7 represents a dictionary. Each path between nodes of the tree represents additional term deletions from the parent node dictionary. Thus a path down any branch of the tree represents a series of successive term deletions (and therefore additions to the negative exclusion list). The root node, N1, represents the initial word list containing all distinct word tokens in the document collection.

Table 6 shows the length of the dictionaries and the total length of the document vectors for each of the dictionaries tested, using the Medlars collection. Fig. 7 and Table 6 should be referred to while reading the following discussion.

RANK	TERM	RELATIVE DISTRIBUTION CORRELATION	DOC. FRQ.	TOT. FRQ.
1	ANESTHETIZED	.5429	4	5
2	SEX	.4653	7	8
3	ANTIBODY	.4611	10	24
4	ANTIGEN	.4545	8	12
5	COLOUR	.4533	8	11
6	WHOM	.4485	9	10
7	FEMALE	.4359	8	11
8	BIRTH	.4199	9	10
9	COLOR	.4146	14	28
10	WHAT	.4049	9	9
11	ANESTHESIA	.4036	11	18
12	EXACT	.3894	7	7
13	THROUGH	.3849	10	10
14	EVER	.3703	11	12
15	INFANT	.3791	18	33
16	THOUGH	.3710	11	11
17	THROUGHOUT	.3695	12	13
18	MALE	.3674	13	18
19	CHILD	.3601	17	24
20	TRUE	.3369	14	14

Terms Ranked According to Relative Distribution Correlation

Table 5


N1
WORD LIST
MANUAL DELETION OF COMMON WORDS
AUTOMATIC DELETION OF FREQUENCY ONE TERMS
SUFFIX 'S' REMOVED
FULL SUFFIXING
CLUSTERING OF TERMS
REMOVAL OF SUFFIX 'S'
M1
WORD FORM
M2
WORD STEM
M3
THESAURUS
A1
DELETION OF LOW FREQUENCY TERMS
MA
STANDARD COMMON WORDS DELETED
NON-DISCRIMINATORS DELETED
HIGH FREQUENCY WORDS DELETED
A2
N2
A4
NON-DISCRIMINATORS DELETED
A3
NON-DISCRIMINATORS DELETED
A5
STANDARD COMMON WORDS DELETED
POOR DISCRIMINATORS DELETED
A6
A5(N)

Dictionaries Used to Test the
Negative Dictionary Construction Methods

Fig. 7

DICTIONARY NAME	DICTIONARY LENGTH	TOTAL LENGTH OF DOCUMENT VECTORS
N1 (WORD LIST)	13,471	84,265
M1 (WORD FORM)	11,142	41,250
M2 (WORD STEM)	9,378	40,200
MA	5,424	37,206
A1	6,226	77,200
A2	5,961	54,200
A3	5,752	36,680
A4	6,196	62,393
A5	5,941	39,142
A6	5,771	36,960

Dictionary Statistics
(Medlars Collection)

Table 6

6.1 Manual Negative Dictionary Construction

Each of the nodes in the left subtree in Fig. 7 (nodes M1, M2, M3, and MA) represents a dictionary formed using *manual* negative dictionary construction methods. The performance results for these dictionaries were presented and discussed in Sections 4.2 and 5.1.1. For convenience, these results are reproduced together in Fig. 8. Automatic negative dictionary construction methods which provide an equivalent or higher level of performance than these manually constructed dictionaries are desired.

6.2 Automatic Negative Dictionary Construction

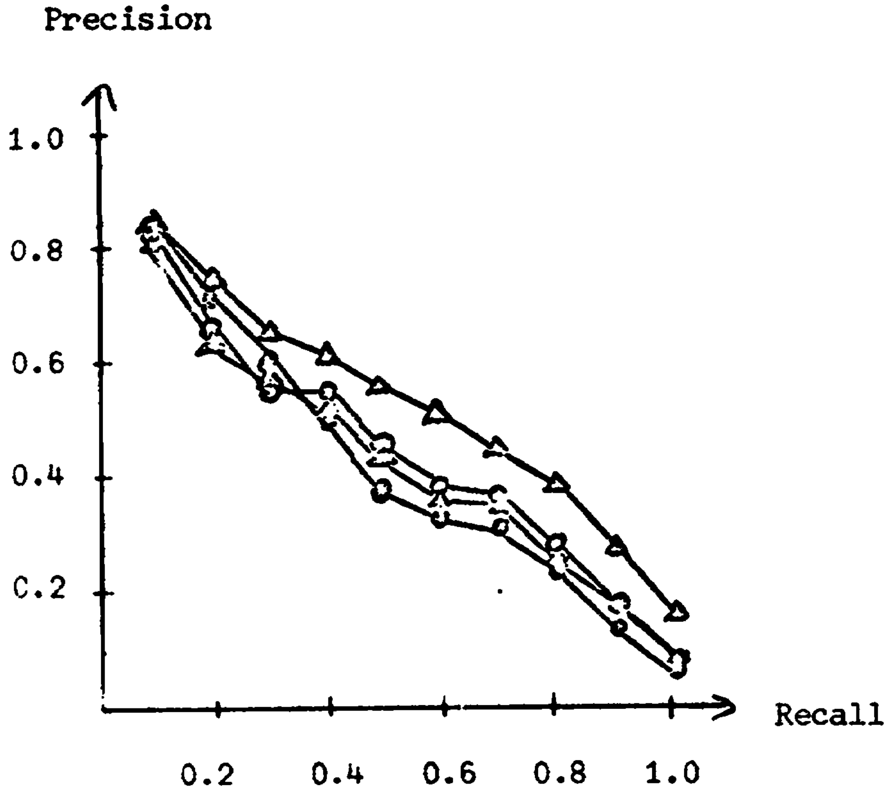
Each of the nodes in the right subtree of Fig. 7 represents a dictionary formed using *automatic* methods. Two automatic operations are performed on the N1 word list to produce the A1 automatic word form dictionary. First, all terms of frequency one are deleted from the N1 dictionary. The deletion of low frequency terms is discussed in Section 5.1.1. It is important to note that of the 7,245 terms which occur only once in the Medlars collection, none occur in any of the queries used. Thus, no word matches between queries and documents are lost due to deletion of these very low frequency terms. Second, the suffix 'S' is removed from all terms. Use of the simple word form dictionary allows accurate comparison of the negative dictionary construction methods tested, without necessitating consideration of other effects on retrieval, as would be necessary if a word stem dictionary or thesaurus were used. A comparison of the performance of the M1-manual word form dictionary and of the A1-automatic word form dictionary is shown in Fig. 9. The performance of the M1 dictionary is superior to that

▲——▲ M1 - Manual word form dictionary

●——● M2 - Manual word stem dictionary

△——△ M3 - Manual thesaurus (Galaska)

○——○ MA - Manual word form dictionary - terms of frequency one deleted



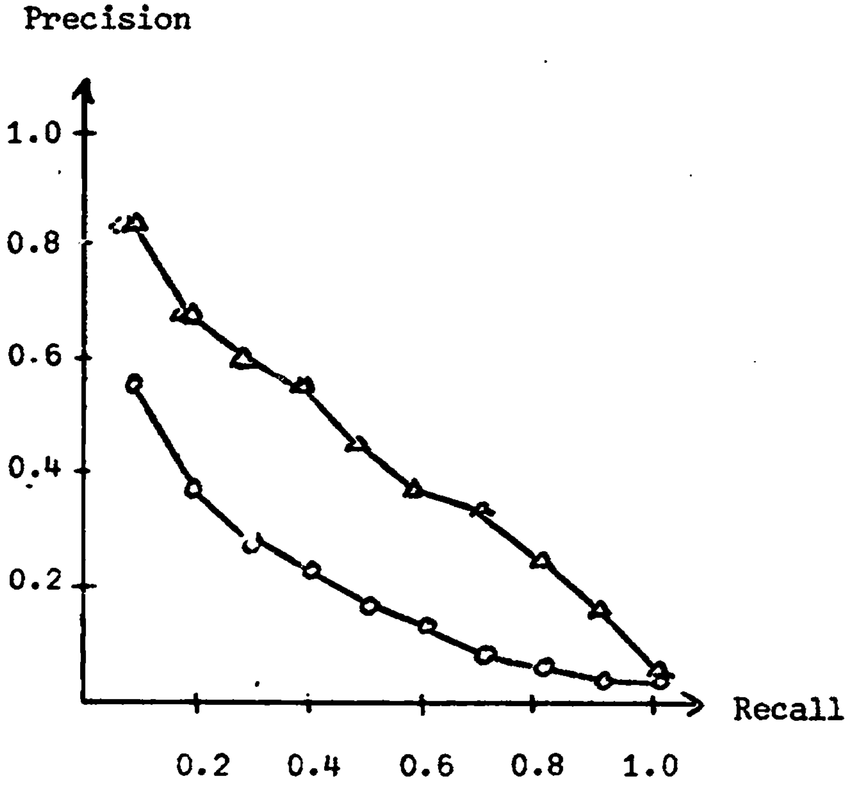
R	Precision			
	▲——▲	●——●	△——△	○——○
0.1	.8253	.8381	.8384	.8276
0.3	.5985	.6198	.6804	.5885
0.5	.4304	.3990	.5151	.4396
0.7	.3252	.3111	.4509	.3253
0.9	.1703	.1338	.2765	.1721

Manual Deletion of Common Words

Fig. 8

△——△ M1 - Manual word form dictionary

○——○ A1 - Automatic word form dictionary - no common words deleted



R	Precision	
	△——△	○——○
0.1	.8253	.5508
0.3	.5985	.2994
0.5	.4304	.1750
0.7	.3252	.0913
0.9	.1703	.0421

Effect of Manual Common Word Deletion

(Medlars Collection)

Fig. 9

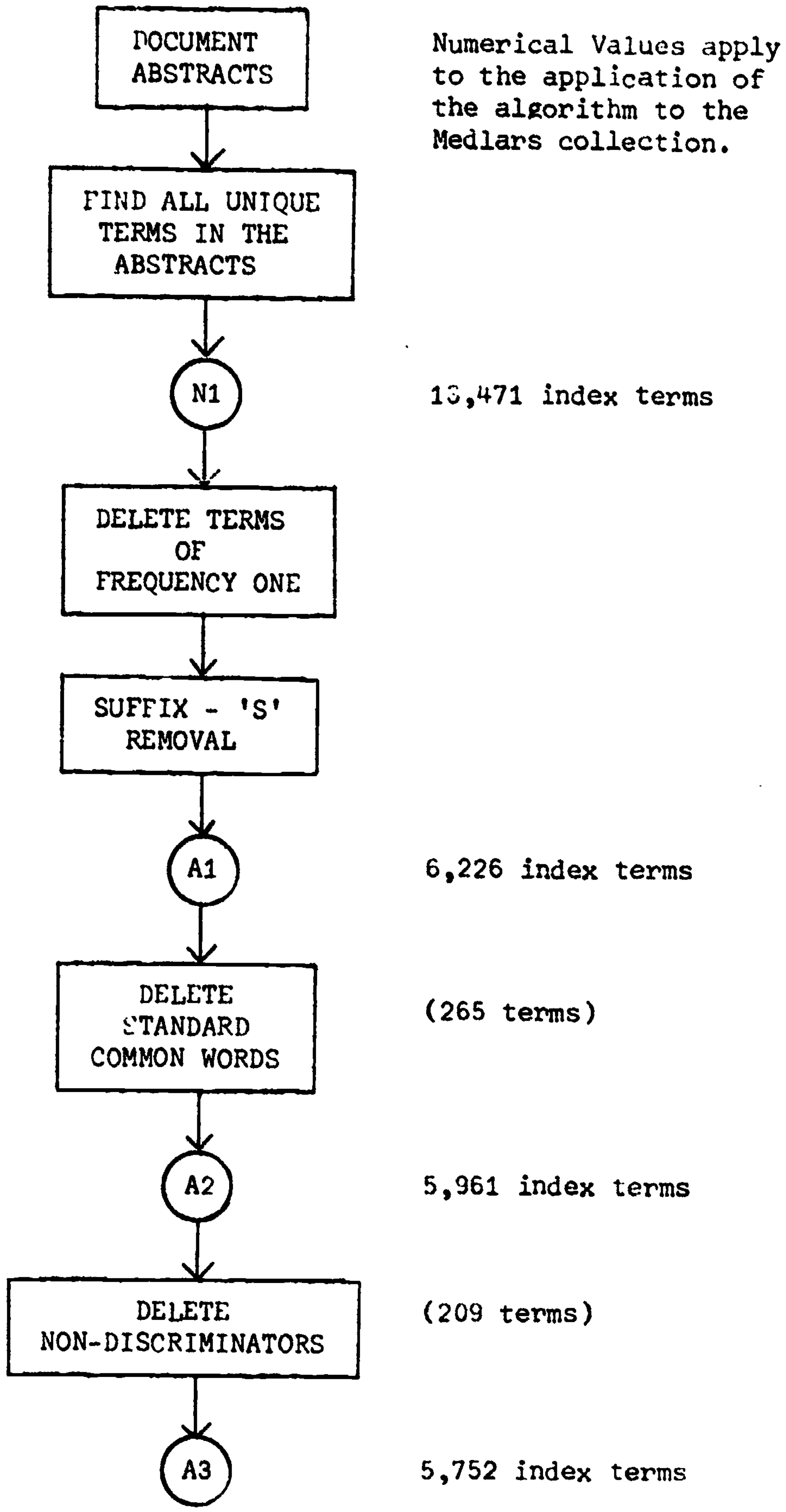
of the A1 dictionary, indicating the effect which manual deletion of common words can have on retrieval performance. However, automatic methods may be used to improve and refine the negative dictionary, increasing the performance of the A1 dictionary to a level above the performance of the M1 dictionary.

As can be seen in Fig. 7, there are four leaf nodes in the subtree with root A1, indicating four procedures used in refining the negative dictionary. These four possible algorithms for negative dictionary construction are considered in the next four sections.

6.2.1 Algorithm P1

Proposed algorithm P1 for negative dictionary construction is outlined in Fig. 10. This algorithm involves use of a standard common word list and the discrimination values of terms. Algorithm P1 closely parallels the manual negative dictionary construction procedure described in Section 4.1. In each case the attempt is to first locate the function words, and then to determine the collection-specific common words. For this purpose, in both the manual and the P1 algorithms, a standard common word list is used to determine the function words. However, in determining collection-specific common words, the manual procedure involves manual examination of word context in a concordance, whereas the P1 algorithm involves use of discrimination value.

Algorithm P1 is not fully automatic due to the necessity of manually constructing the standard common word list. However, this list may be constructed only once and retained as a data set for repeated use with each new collection processed. The use of a standard common word list may therefore



Numerical Values apply to the application of the algorithm to the Medlars collection.

Algorithm P1

For Negative Dictionary Construction

Fig. 10

l considered as a semi-automatic method. For the Medlars collection, there are 256 words in the A1 dictionary which occur on the standard common word list. These words are therefore deleted from the A1 dictionary, producing the A2 dictionary.

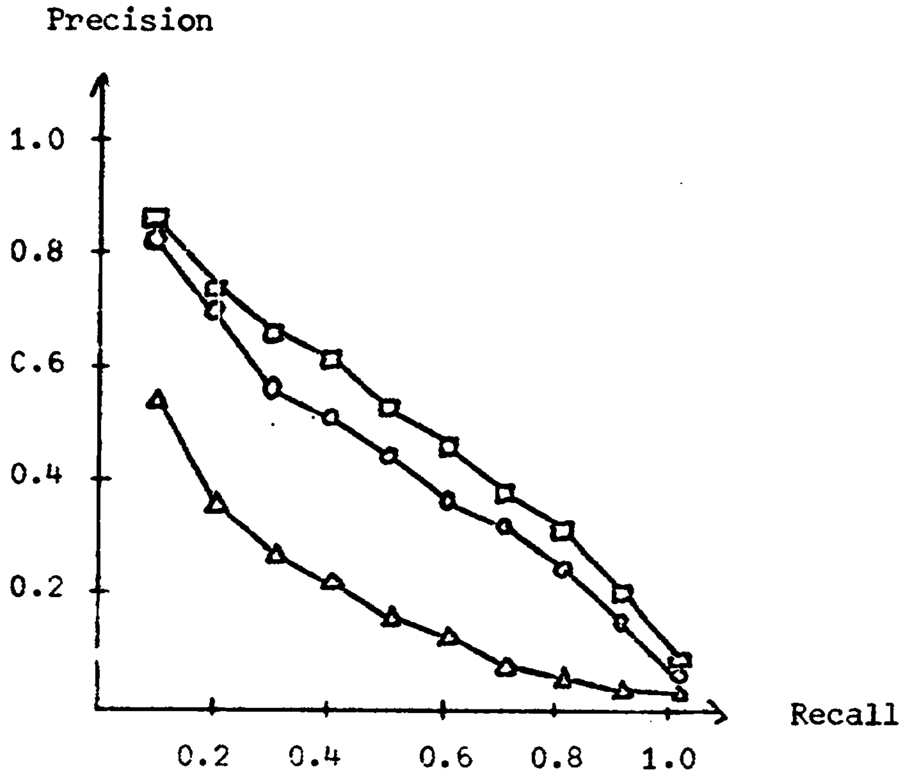
Using the document vectors derived from the A2 dictionary, the discrimination values for the 5,961 terms in the A2 dictionary are computed. There are 209 terms which are non-discriminators and these are deleted from the A2 dictionary to produce the A3 dictionary. Fig. 11 shows the performance curves of the A1, A2, and A3 dictionaries. The improved performance of the dictionaries produced at successive steps of this algorithm is apparent. The importance of recognizing and deleting function words is emphasized by the improvement between the A1 and A2 dictionaries. However, the additional improvement of the A3 dictionary shows the importance of determining collection-specific common words as well, and indicates the usefulness of discrimination value in doing so.

The effectiveness of this semi-automatic algorithm for negative dictionary construction is judged by comparison with manual methods. Fig. 12 shows the performance of the A3 dictionary in comparison to that of the M1 manual word form dictionary. The M1 dictionary performed slightly better than the A3 dictionary at the high recall values of .95 and 1.00. At all other recall points, the A3 dictionary performed distinctly better than the M1 dictionary. The conclusion may be drawn that semi-automatic algorithm P1 may be used to produce dictionaries which perform as effectively as manually constructed dictionaries in information retrieval.

△——△ A1 - Freq one terms deleted

○——○ A2 - Freq one and standard common words deleted

□——□ A3 - Freq one, standard common, and non-discriminators deleted



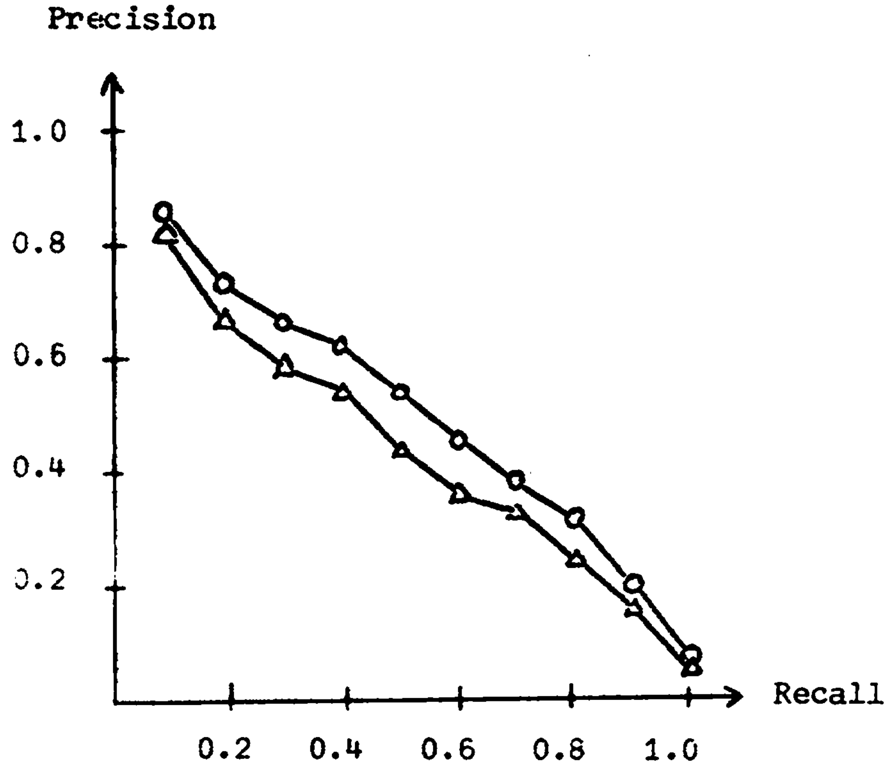
R	Precision		
	△——△	○——○	□——□
0.1	.5508	.8260	.8436
0.3	.2994	.5745	.6659
0.5	.1750	.4392	.5213
0.7	.0913	.3187	.3776
0.9	.0421	.1591	.2037

Automatic Deletion of Common Words

Fig. 11

△——△ M1 - word form, manual common word deletion

○——○ A3 - Freq one terms, standard common words, and non-discriminators deleted automatically



R	Precision	
	△——△	○——○
0.1	.8253	.8436
0.3	.5985	.6659
0.5	.4304	.5213
0.7	.3252	.3776
0.9	.1703	.2037

Comparison ~: Manual and Automatic

Commcn Word Deletion

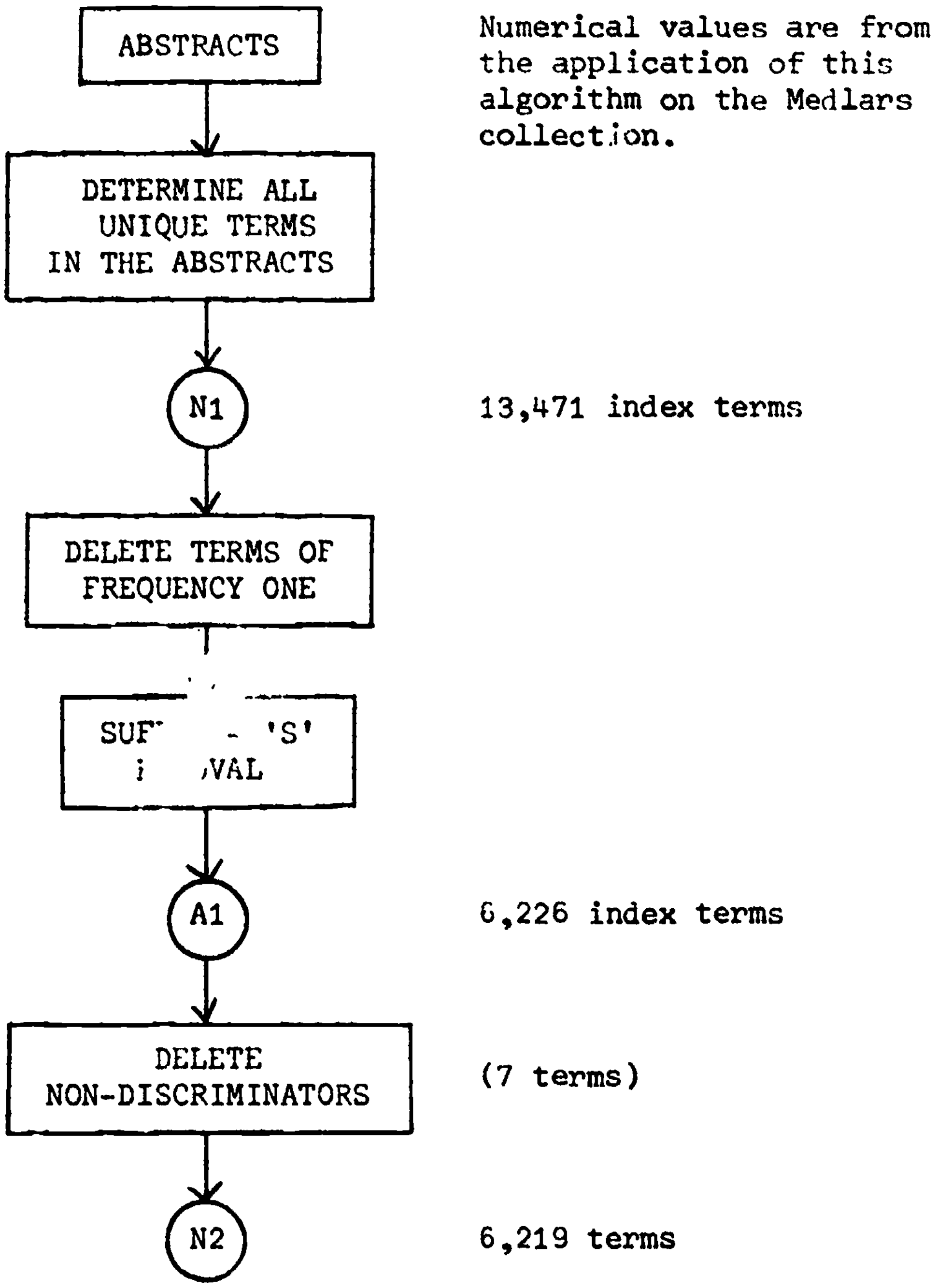
Fig. 12

6.2.2 Algorithm P2

Algorithm P2, which is shown in Fig. 13, involves a single automatic refinement of the A1 dictionary. Testing of this algorithm on the Medlars collection shows that it is not a useful algorithm for automatic negative dictionary construction. The reason for this is of interest, particularly as it provides insight into the discrimination value function.

Computation of the discrimination values for each of the 6,226 terms in the A1 dictionary shows that only 7 terms are non-discriminators. These 7 non-discriminators each occur in over 73% of the documents, and their total combined frequency of 37,875 accounts for over 25% of the total occurrence of all terms in the collection. Thus, these are very high frequency terms and they have a dominating effect on the calculations necessary to compute discrimination values. In a sense, the collection is *stable* with respect to these seven terms. Deletion of any other single term has only a very minor effect on the compactness of the collection due to the dominance of these seven high frequency terms. Since the A1 dictionary contains no terms of very low frequency (i.e. frequency one), the average frequency of all the terms is much higher than in the N1 word list. This probably is what causes the poor results produced by the P2 algorithm. It may be advisable then, to delete very high frequency terms to offset the effect of deleting very low frequency terms.

The results here show that when the average frequency of the terms in a collection is greatly shifted, the discrimination values computed for the remaining terms may not give a true indication of which terms are


ABSTRACTS
Numerical values are from the application of this algorithm on the Medlars collection.
DETERMINE ALL UNIQUE TERMS IN THE ABSTRACTS
N1
13,471 index terms
DELETE TERMS OF FREQUENCY ONE
SUF - 'S'
VAL
A1
6,226 index terms
DELETE NON-DISCRIMINATORS
(7 terms)
N2
6,219 terms

Algorithm P2

For Negative Dictionary Construction

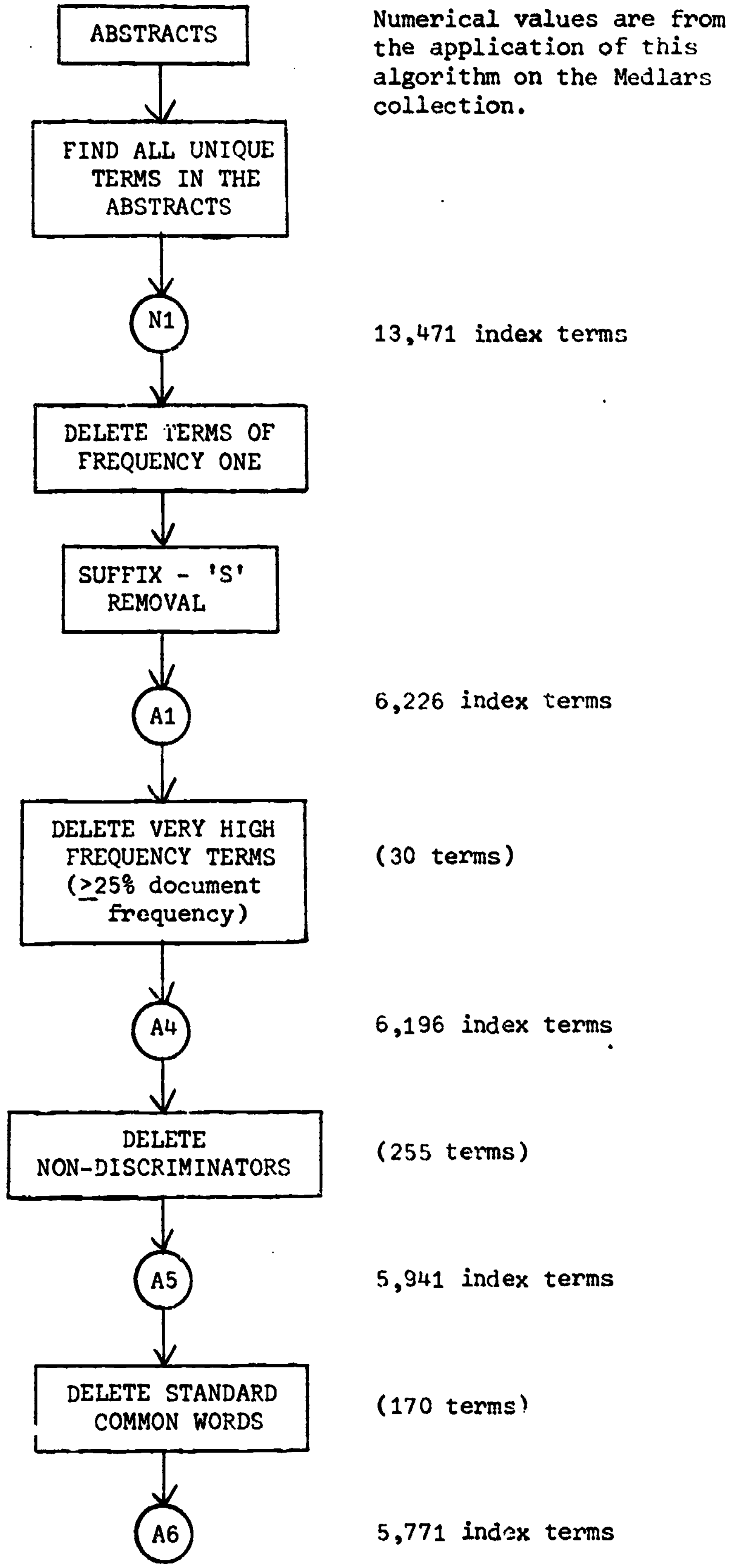
Fig. 13

non-discriminators in the original collection. The algorithm discussed in the next section provides a solution to this difficulty.

6.2.3 Algorithm P3

Algorithm P3 for negative dictionary construction is shown in Fig. 14. The A1 dictionary is refined first ' deletion of all terms occurring in 25% or more of the documents, as discussed in Section 5.1.2. There are 30 such high frequency terms (see Table 2, Section 5.1.2) and these are deleted from the A1 dictionary to produce the A4 dictionary. Discrimination values are computed for the 6,196 terms in the A4 dictionary. Some of the terms which are non-discriminators, along with some of the best discriminators in the collection are listed in Table 7. Altogether there are 255 non-discriminators in the A4 dictionary, and these are deleted, forming the A5 dictionary. It should be clear upon examination of each step in Fig. 14 that the A5 dictionary is a fully automatically constructed dictionary. Essentially three values are specified in executing this algorithm. These are: the low frequency cutoff value, the high frequency cutoff value, and the discrimination cutoff value. The choice of these values has already been discussed; however, in the case of discrimination cutoff value, several values are considered and tested in the next section.

Examination of the A5 dictionary shows that there are 170 words in it which are also on the standard common word list. It is intuitive that deletion of these function words from the A5 dictionary may increase its level of performance in retrieval. To test this, the A6 dictionary is constructed, by deleting all terms from the A5 dictionary which occur on the standard common word list.



Numerical values are from the application of this algorithm on the Medlars collection.

ABSTRACTS
FIND ALL UNIQUE TERMS IN THE ABSTRACTS
N1
13,471 index terms
DELETE TERMS OF FREQUENCY ONE
SUFFIX - 'S' REMOVAL
A1
6,226 index terms
DELETE VERY HIGH FREQUENCY TERMS ($\geq$25% document frequency)
(30 terms)
A4
6,196 index terms
DELETE NON-DISCRIMINATORS
(255 terms)
A5
5,941 index terms
DELETE STANDARD COMMON WORDS
(170 terms)
A6
5,771 index terms

Algorithm P3
For Negative Dictionary Construction
Fig. 14

NON-DISCRIMINATORS				DISCRIMINATORS			
Term	Document Frequency	Total Frequency	Average Frequency	Term	Document Frequency	Total Frequency	Average Frequency
1. CELL	208	785	3.77	1. DNA	43	201	4.67
2. CASE	253	521	2.06	2. ANTIGEN	34	128	3.76
3. HAVE	256	332	1.30	3. NICKEL	21	78	3.71
4. EFFECT	207	346	1.67	4. HGH	11	46	4.18
5. MAY	222	327	1.47	5. AMYLOIDOSI (S)	18	73	4.06
6. NORMAL	203	369	1.82	6. TUMOUR	24	66	2.75
7. TREATMENT	181	300	1.66	7. HEPATITI (S)S	13	49	3.77
8. HAS	225	305	1.36	8. OXYGEN	31	104	3.35
9. RESULT	227	285	1.26				
10. OTHER	234	304	1.30				

Highest Ranking Discriminators and Non-Discriminators

(Medlars Collection of 1033 Abstracts)

Table 7

The performance curves for the A1, A4, A5, and A6 dictionaries are given in Fig. 15. Possibly the most surprising result is the performance of the A4 dictionary compared to the A1 dictionary, as these dictionaries differed by only 30 high frequency terms. The A4 dictionary represents the performance which is obtained by simple deletion of very high and very low frequency terms. The deletion of non-discriminators to form the A5 dictionary results in another significant increase in performance level. However, no significant gain results from further refinement of the A5 to produce the A6 dictionary. Thus, it may be concluded that the final semi-automatic step of deleting standard common words is not warranted. The suggested P3 algorithm is therefore concluded with the formation of the A5 dictionary.

The result of this algorithm P3 (the A5 dictionary) is compared to the results of the algorithm P1 (the A3 dictionary) and the manual algorithm (the M1 dictionary) in Fig. 16. The performance of the A5 dictionary is distinctly better than that of the M1 dictionary, and slightly improved from the performance of the A3 dictionary. The fully automatic algorithm for negative dictionary construction, P3, is therefore superior to either the manual algorithms now in use, or the semi-automatic algorithm, P1.

6.2.4 Algorithm P4

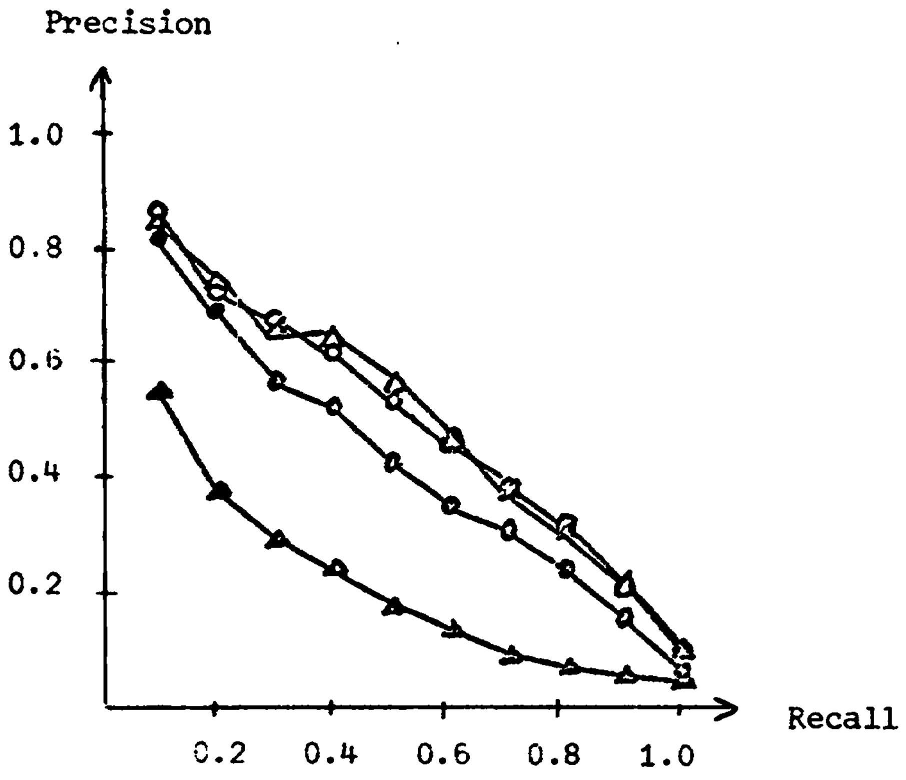
In Section 5.2.1 the idea of a discrimination cutoff value, D_c was suggested, and the idea of the classification of terms into three areas; non-discriminators, poor discriminators, and good discriminators, was discussed. In the previous algorithms, only non-discriminators were deleted, that is, a

▲—▲ A1 - Freq one terms deleted

●—● A4 - Freq one and high freq term: deleted

△—△ A5 - Freq one, high freq, and non-discriminators deleted

○—○ A6 - Freq one, high freq, non-discriminators, and standard common words deleted



R	Precision			
	▲—▲	●—●	△—△	○—○
0.1	.5508	.8264	.8304	.8415
0.2	.2994	.5779	.6689	.6716
0.5	.1750	.4153	.5332	.5246
0.7	.0913	.3026	.3717	.3768
0.9	.0421	.1501	.2022	.2023

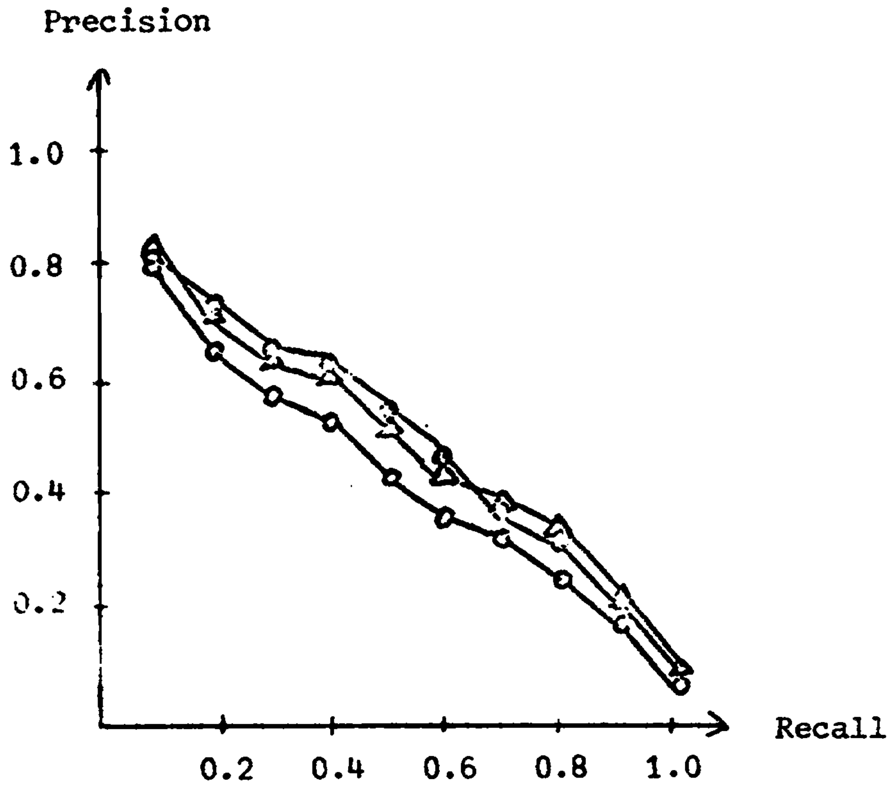
Automatic Deletion of Common Words

Fig. 15 .

○—○ M1 - Word form, manual common word deletion

△—△ A3 - Automatic deletion of freq one terms, standard common words, and non-discriminators

●—● A5 - Automatic deletion of freq one terms, high freq terms, and non-discriminators



R	Precision		
	○—○	△—△	●—●
0.1	.8253	.8436	.8304
0.3	.5985	.6659	.6689
0.5	.4304	.5213	.5332
0.7	.3252	.3776	.3717
0.9	.1703	.2037	.2022

Comparison of Manual and Automatic

Common Word Deletion

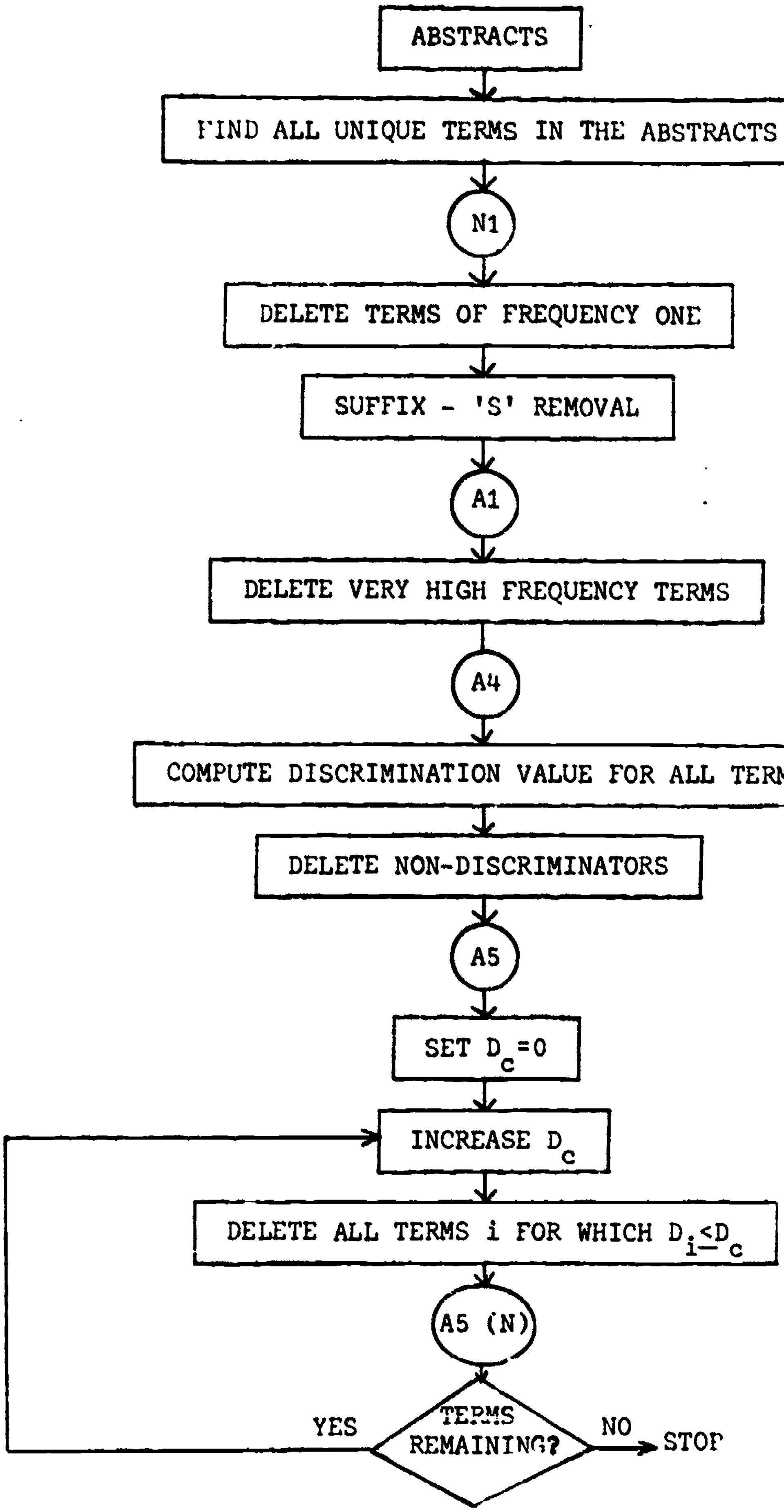
Fig. 16

discrimination cutoff value of zero was chosen. Using the P4 algorithm shown in Fig. 17 the effect of deleting poor discriminators in addition to non-discriminators is tested. This algorithm is identical to the P3 algorithm discussed in the last section in its construction of the A5 dictionary. In the P4 algorithm, however, groups of terms are deleted from the A5 dictionary in increasing order of discrimination value, that is, the discrimination cutoff value is gradually increased.

Table 8 shows the performance values[(iii)] for the dictionaries resulting from deletion of an increasing number of terms. These dictionaries are listed in decreasing order of the number of terms remaining in the dictionary (i.e. the number of terms to be used in indexing the documents and queries). Thus, the A5 (5940) is the original A5 dictionary with all 5940 terms retained, while the A5 (250) is the A5 dictionary with only the 250 best discriminators remaining. The results shown in Table 8 are displayed graphically in Fig. 18. The results for the A1 dictionary are included in both Table 8 and Fig. 18 for purposes of comparison.

Analysis of Fig. 18 shows that in fact the classification of terms as good discriminators, poor discriminators, and non-discriminators is reasonable. The points at the right end of Fig. 18 show the large improvement in performance which occurs upon the deletion of non-discriminators. However, moving across the graphs from right to left, very

(iii) These measures are described in [16].


ABSTRACTS
FIND ALL UNIQUE TERMS IN THE ABSTRACTS
N1
DELETE TERMS OF FREQUENCY ONE
SUFFIX - 'S' REMOVAL
A1
DELETE VERY HIGH FREQUENCY TERMS
A4
COMPUTE DISCRIMINATION VALUE FOR ALL TERMS
DELETE NON-DISCRIMINATORS
A5
SET $D_c=0$
INCREASE D_c
DELETE ALL TERMS i FOR WHICH $D_i \le D_c$
A5 (N)
TERMS REMAINING?
YES
NO
STOP

Deletion of poor discriminators, for increasing discrimination cutoff values.

Algorithm P4
For Negative Dictionary Construction
Fig. 17

DICTIONARY (TERMS RETAINED)	NORM RECALL	NORM PRECISION	NORM RECALL	NORM PRECISION
A1 (6225)	.7265	.5021	.0606	.4991
A5 (5940)	.7841	.7260	.1450	.6594
A5 (5709)	.7819	.7253	.1448	.6591
A5 (5459)	.7819	.7250	.1440	.6580
A5 (5209)	.7819	.7250	.1441	.6581
A5 (4960)	.7819	.7245	.1439	.6575
A5 (4712)	.7819	.7245	.1439	.6576
A5 (4212)	.7800	.7228	.1437	.6562
A5 (3715)	.7786	.7213	.1439	.6552
A5 (2717)	.7741	.7212	.1430	.6544
A5 (2000)	.7654	.7149	.1337	.6500
A5 (1500)	.7625	.7135	.1343	.6490
A5 (1000)	.7525	.7066	.1192	.6425
A5 (500)	.7337	.6466	.0982	.5883
A5 (250)	.6586	.4353	.0640	.4610

Retrieval Performance Upon
Successive Reduction of Index Terms by
Increasing Discrimination Cutoff Value
(Medlars Collection)

Table 8

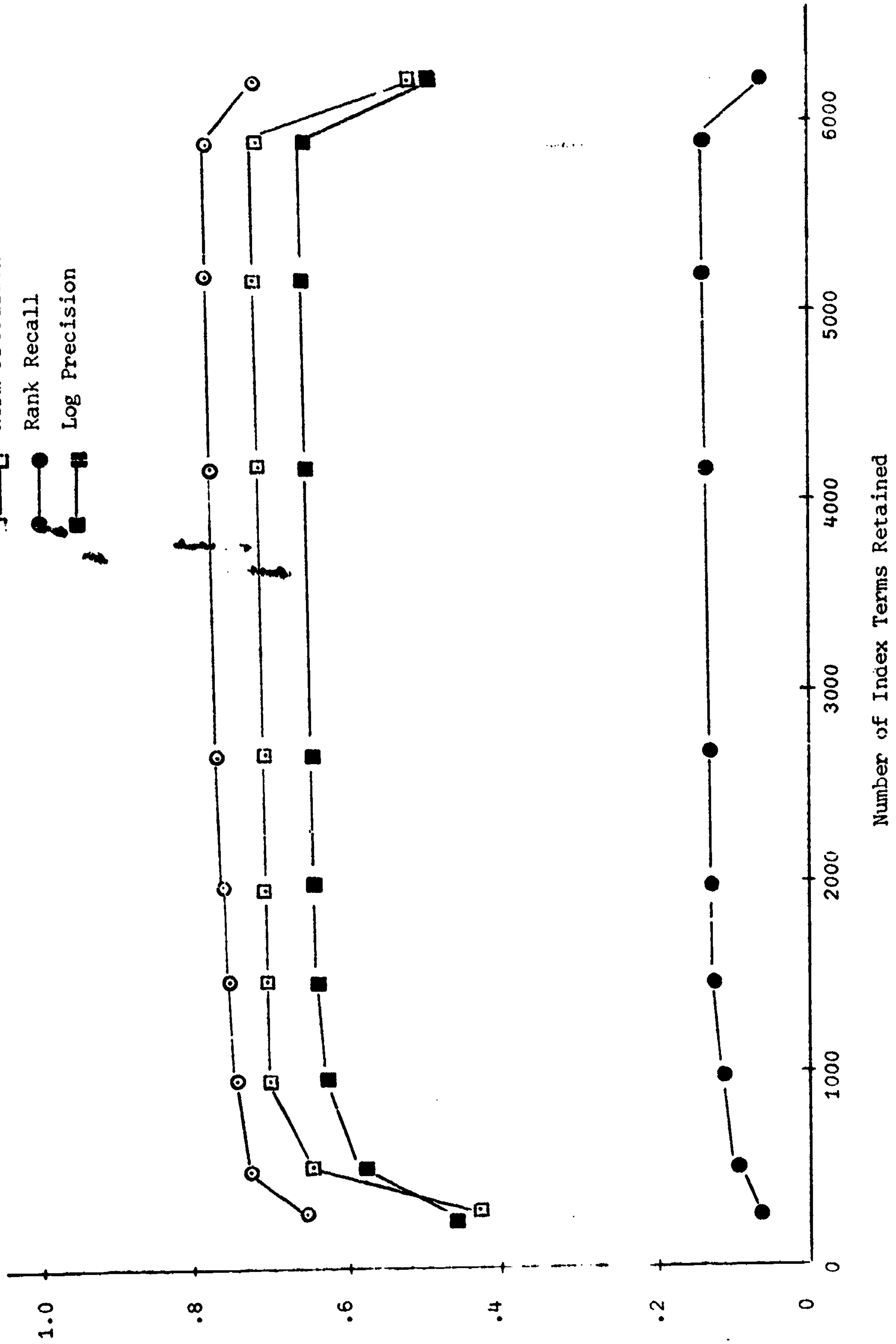

Norm Recall
Norm Precision
Rank Recall
Log Precision
1.0
.8
.6
.4
.2
0
0
1000
2000
3000
4000
5000
6000
Number of Index Terms Retained

Fig. 18

little change is noticed in any of the measures between the points for 5940 and 1000 index terms. Thus, almost 5000 terms are present in the collection which have little effect on retrieval results. These are the poor discriminators. From the point at which 1000 terms remain in the dictionary, further deletion of terms results in a decrease in each of the retrieval measures. Thus, these last 1000 terms, having the highest discrimination values, are good discriminators.

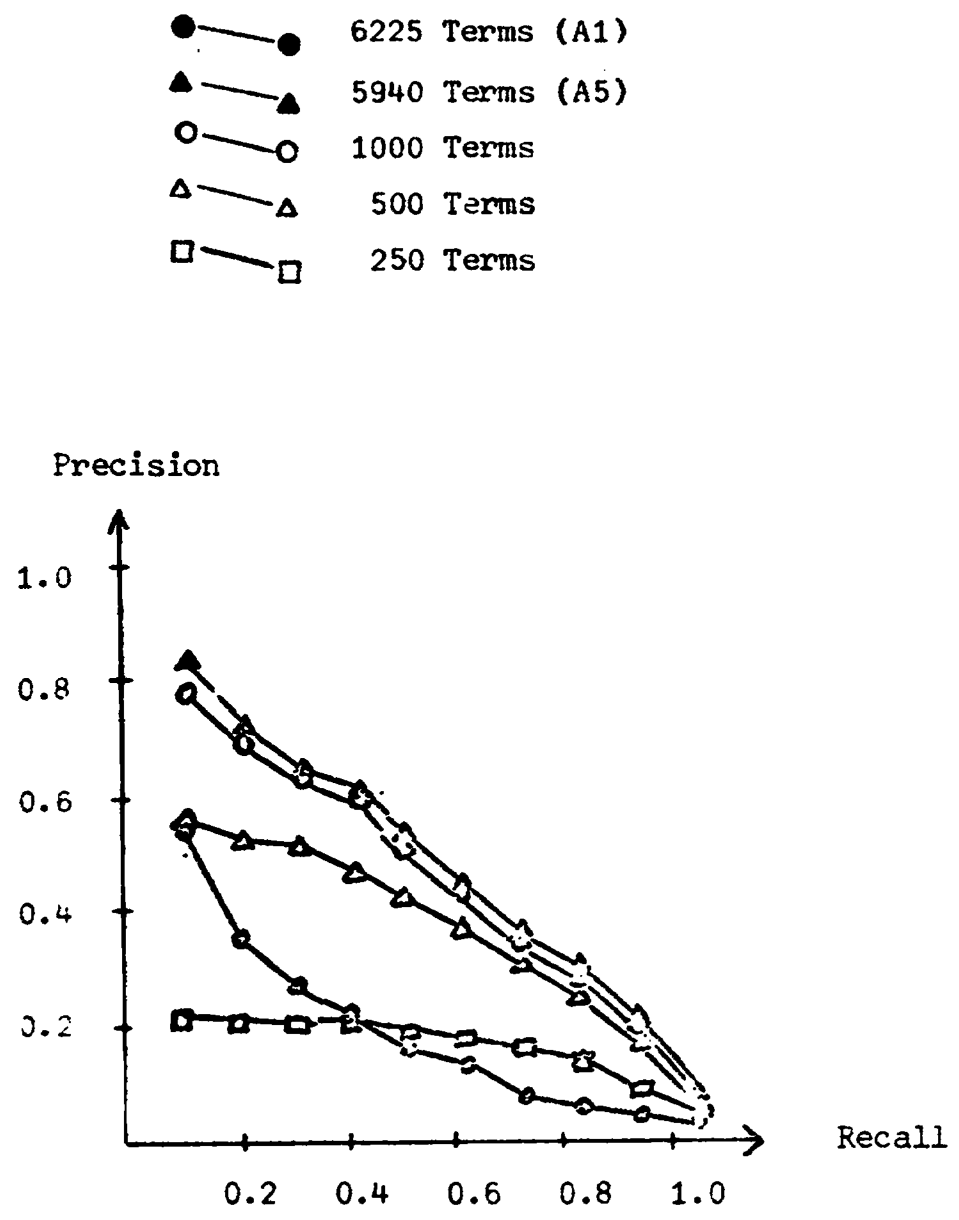
These results are clarified further by the display in Fig. 19 of performance curves for several of the dictionaries. The results shown in Fig. 19 may be summarized as follows:

(i) Deletion of non-discriminators results in a significant increase in performance level (the change from A1 to A5 (5940)).

(ii) Deletion of poor discriminators has the effect of decreasing performance level very slightly (the change from A5 (5940) to A5 (1000)).

(iii) Deletion of good discriminators casues a sharp decrease in performance level (the changes from A5 (1000) to A5 (500) to A5 (250)).

Whether or not poor discriminators are deleted from the dictionary depends on the constraints of the particular retrieval environment under consideration. If recall is to be maximized, then the poor discriminators must not be deleted. However, if a small decrease in recall may be tolerated, then poor discriminators may be deleted, resulting in a significant decrease in dictionary size.



R	Precision				
	●—●	▲—▲	○—○	△—△	□—□
0.1	.5508	.8304	.7939	.5531	.2145
0.3	.2994	.6689	.6670	.5307	.2108
0.5	.1750	.5332	.5126	.4286	.1979
0.7	.0913	.3717	.3661	.3292	.1793
0.9	.0421	.2022	.2017	.1813	.0925

Comparison of Retrieval Results as the
Number of Discriminators is Reduced

Fig. 19

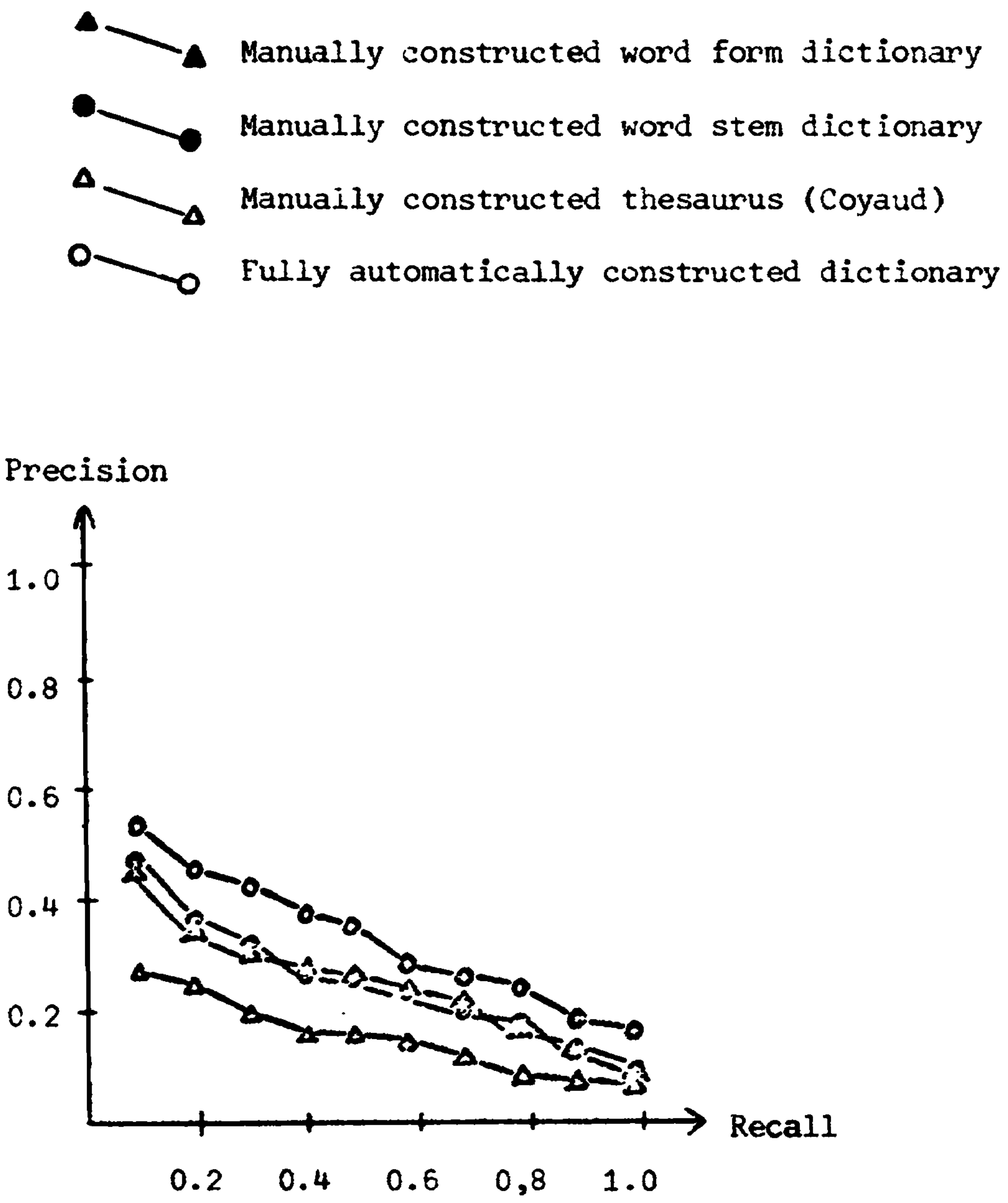
BEST COPY AVAILABLE

6.2.5 Conclusions and Verification

On the basis of the results presented, it is concluded that fully automatic methods may be used for constructing negative dictionaries which provide for a higher level of retrieval performance than do standard manual methods. In particular, the P4 algorithm is an efficient and effective algorithm for use in negative dictionary construction. In addition, it has been found that a dictionary of index terms may be greatly reduced in size by deletion of poor discriminators without affecting retrieval performance significantly.

Since these conclusions are based on tests performed on a single collection, it is worthwhile to verify the results using another document collection. The Ophthalmology collection is used, with results being compared for three manually constructed dictionaries and one automatically constructed dictionary. The manually constructed dictionaries are a word form dictionary, a word stem dictionary, and a thesaurus. The automatically constructed dictionary is the A5 dictionary, constructed according to algorithm P4. There are 3762 terms (discriminators) in this A5 dictionary. Since there is adequate space available in the present utility for handling this dictionary, no deletion of poor discriminators is necessary.

The performance of the manual word form, manual word stem, manual thesaurus, and automatic A5 dictionaries is shown in Fig. 20. The A5 dictionary performs better at all recall values than each of the manually constructed dictionaries. Thus, the conclusion that automatic negative dictionary construction methods are effective, is substantiated.



R	Precision			
	▲——▲	●——●	△——△	○——○
0.1	.4349	.4380	.2737	.5185
0.3	.3004	.3019	.1991	.4074
0.5	.2665	.2656	.1541	.3420
0.7	.2024	.1988	.1052	.2548
0.9	.1268	.1263	.0733	.1848

Comparison of Manual and Automatic Methods
of Dictionary Construction (Ophthalmology Collection)

Fig. 20

7. Summary and Conclusions

Negative dictionary construction algorithms are described in this study that use manual, semi-automatic, and fully automatic methods to determine common words. The intention is to show that dictionaries constructed by fully automatic methods perform equivalent to or better than dictionaries formed manually. Experimental evidence from two document collections indicates that automatic negative dictionary construction methods are more effective than standard manual negative dictionary construction methods, and are therefore to be preferred.

In addition to the general results regarding negative dictionary construction, important results are found regarding the discrimination value function. The usefulness of discrimination value is verified both theoretically and experimentally, and its properties are more fully explored. The relationships of words in a collection are examined, in terms of frequency, distribution, and discrimination value. Finally, experimental results are given which show that dictionary size may be greatly reduced retaining only good discriminators, while maintaining retrieval performance.

References

[1] H.P. Luhn, IBM J. Res. Develop. 1, No. 4, 1957.

[2] ________, "Potentialities of Auto-Encoding of Scientific Literature", IBM Res. Cent. Rep. No. RC-101, 1959.

[3] G. Salton, "Automatic Text Analysis", Science, Vol. 168, April 1970.

[4] C.W. Cleverdon and E.M. Keen, "Factors Determining the Performance of Indexing Systems", Aslib Cranfield (England) Res. Proj. Rep. Vols. 1 and 2, 1966.

[5] G. Salton, "A Comparison Between Manual and Automatic Indexing Methods", American Documentation, Vol. 20, No. 1, January 1969.

[6] D. Bergmark, "The Effect of Common Words on Retrieval Performance", Scientific Report No. ISR-18 to the National Science Foundation and to the National Library of Medicine, Department of Computer Science, Cornell University, October, 1970.

[7] D. Williamson, R. Williamson, and M. Lesk, "The Cornell Implementation of the SMART System", Report No. ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

[8] G. Salton, editor, The SMART Retrieval System (Prentice Hall, New Jersey, 1971).

[9] ________, et al., "Information Storage and Retrieval", Department of Computer Science, Cornell University, Rep. Nos. ISR-11, ISR-12, ISR-13, ISR-14, ISR-15, ISR-16, ISR-17, ISR-18, ISR-19 to the National Science Foundation (1966-1971).

[10] J.J. Rocchio, "Performance Indices for Document Retrieval Systems", Report No. ISR-8 to the National Science Foundation, Harvard University, December 1964.

[11] P.K.T. Vaswani and J.B. Cameron, "Statistical Word Associations and Their Use in Document Indexing and Retrieval", National Physical Laboratory, Com. Sci. 42, April 1970.

[12] G. Salton and M.E. Lesk, "Information Analysis and Dictionary Construction", Report No. ISR-11 to the National Science Foundation, Department of Computer Science, Cornell University, June 1966.

[13] G. Salton, Automatic Information Organization and Retrieval (McGraw-Hill, New York, 1968).

[14] J. Aste-Tonsman and K. Bonwit, "Negative Dictionaries", Report No. ISR-18 to the National Science Foundation and to the National Library of Medicine, Department of Computer Science, Cornell University, October, 1970.

[15] C. Yu and A. Wong, "Modification of the Document Space Similarity", Project Report, Department of Computer Science, Cornell University, June 1971.

[16] G. Salton and M.E. Lesk, "Computer Evaluation of Indexing and Text Processing", Report No. ISR-12 to the National Science Foundation, Department of Computer Science, Cornell University, June, 1967.

Dynamically versus Statically Obtained Information Values

A. van der Meulen

Abstract

An evaluation is made of the effectiveness of dynamically updated parameters, known as "information values", and a comparison is made with a statistical approach in which parameter values are computed once rather than being continually changed.

Information values are quantities which may be assigned to dictionary items in order to reflect the descriptive power of the various keywords. The main objective of the usage of information values is the improvement of system performance by taking into account the usefulness of the index terms in content analysis. Simultaneously a means of control over the index vocabulary is obtained, since an information value displays the quality of its associated keyword.

The procedure of assigning information values is based on the philosophy that keywords which help to retrieve relevant documents are more valuable than those which participate in retrieving nonrelevant documents. In particular the utilization of user relevancy decisions is examined in two different ways. First, the collected data reflecting the retrieval history of a system is used to compute information values in a rather statistical

way. Second, each retrieval result is used on its own to change and update the current information values.

1. Introduction

The investigations described in t'.is report may be regarded as an extension of earlier work done with information values [1]. It is suggested there that a realistic dynamic updating be carried out rather than the simplified simulation used earlier which will be referred as the "static" updating approach. The process of using the retrieval results of each previous search to improve the results of the next will be called dynamic updating.

A known exampl· of dynamic system updating is the "dynamic document modification procedure". [2] Imperfect document indexes can be improved by utilizing the retrieval results as evaluated by the user (feedback). After each search the indexes of the retrieved documents are slightly changed in such a way that a new user who wants similar information will find his relevant documents ranked higher and his retrieved nonrelevant documents ranked lower. One might wonder whether this process of continuous index modifications is stable in that finally, after intensive system use, the system performance reaches an optimum; this question has not been investigated yet for dynamic document space modification.

Another type of dynamic updating is proposed for the so-called information values, quantities which give a numeric value for the quality (discrimination power) of the identifiers (keywords) used in the indexing process. In previous experiments [1] with these values a statistical approach is used for practical reasons. It differs from the dynamic strategy in that the retrieval results are stored for a whole set of queries from which in turn the information values are computed. In the dynamic approach, however, the retrieval results for each individual query are affected by the results obtained for the previous one, since the updating occurs continuously (after each search).

2. Information Values and their Derivation

A. The Concept

In regular relevance feedback, the user judges the retrieved documents as either relevant or not relevant to his query. This information is then fed back into the system and used to redefine the query for subsequent search iterations. These relevance judgments may also be used automatically to construct a weighted dictionary, as is explained in the next section.

Specifically, with each identifier one can associate a numerical value, herein referred to as the information value. This information value is initially set equal to one, and is changed only when the corresponding concept occurs both in the query and in a retrieved document. The information value is then increased if that retrieved document is declared relevant, or else it is decreased. In the indexing process, these information values may be used in addition to the existing identifier frequency weights; that is, each vector-weight is formed as the product of both.

This will result in document and query vectors in which the prominent descriptors are emphasized, while the lesser ones are suppressed.

The whole procedure is simple to implement. An attempt is thus made to determine whether the process leads to an improvement in system performance — that is, a lifting of the recall-precision curve.

B. The Updating Method

Initially, the information value of each concept is set equal to one. From that point the value is increased (or decreased) each time the corresponding concept co-occurs both in a query and the relevant (or nonrelevant) documents retrieved in response to that query.

The specific increment-decrement function chosen is the one proposed by Sage [3], herein referred to as the "sine-function" because of its resemblance to the regular sine. If i is an identifier which co-occurs both in a query and in its associated retrieved document, define:

v_i = the information value of identifier i (initially set equal to one)

v_i^*= the information value of identifier i after updating

x_i = arc sin (v_i - 1), the transposed information value.

Then, $v_i = 1 + \sin (x_i)$

and similarly v_i^* is calculated by

$$v_i^* = 1 + \sin (x_i \pm \Delta x_i) \qquad (1)$$

where Δx is a function of the old information value, calculated as

$$\Delta x_i = \frac{\pi/2 - |x_i|}{C} \qquad (2)$$

where C is an arbitrary constant, set eqaul to 8 in these experiments.

Δx_i is added in equation (1) if the retrieved document containing term i is judged relevant by the user; or it is subtracted in the information value calculation if the corresponding document is judged nonrelevant.

C. Dynamic versus Static Updating

The information values are increased as long as their identifiers occur in relevant retrieved documents, but are decreased in response to nonrelevant retrieval. Each new query is transformed by the weighted dictionary in the following way:

$$Q = (v_1w_1, v_2w_2, \ldots, v_nw_n) \qquad (3)$$

where w_i is the regular frequency weight of identifier i and v_i the corresponding information value. Notice also that new incoming documents will be indexed in a similar way, using the weighted dictionary.

An intuitive explanation for the existence of an equilibrium value follows. Consider again identifier i and the corresponding information value v_i. The similarity measure between the query and the document vector is computed as the cosine of the angle between them. As a result of each relevant retrieval v_i is updated. Consequently the term yields a greater influence on the direction of the corresponding vector in the index-space. Since identifier i is emphasized, more documents containing that identifier are retrieved when a larger correlation coefficient is obtained.

As long as this promoted identifier is successful in retrieving relevant documents, the corresponding information value will be increased. If, however, nonrelevant documents are retrieved because of an excessively high value for a given concept, then the self-correcting feedback mechanism comes into play and the information value will be set back again.

The assumption is that in this way each identifier eventually achieves an equilibrium value which is characteristic of its descriptive power.

The mechanism described above defines a dynamic dictionary update procedure, wherein the dictionary is modified after each query search. In this situation each query is indexed only after the dictionary has been altered in response to all previously searched queries.

The first experiment with information values uses labor saving batching techniques. The document collection is searched by a batch of queries indexed with the unmodified dictionary. Sufficient information is stored to allow updating the information values on the basis of the two highest ranked retrieved documents associated with each query.

This procedure, however, may exaggerate the information values obtained, since the self-correcting capacity of the feedback process described earlier is never invoked. Specifically, a good or bad term will be continually changed based on its retrieval effectiveness using its initial information value, rather than its retrieval effectiveness using its perhaps modified information value. Nevertheless, since a relatively small number of updates normally occurs, it seems plausible that these values may still give a good indication of the information value of the associated identifiers.

It is interesting to draw a comparison between dynamic and static updating; the more realistic dynamic approach is simulated within the SMART system. Details of the dynamic experiments are described in the next section.

3. The Experiments

Information values are changed according to an algorithm which takes into account the co-occurrence of concepts in the retrieved documents and the associated query. Information values are increased if the retrieved documents are relevant and decreased otherwise. The dynamic updating is done immediately after each query of the query update collection has been searched. Updated information values are used in indexing the next query, and the retrieval results again modify the information values.

For the purpose of comparing the static and the dynamic approach, two dynamic experiments are carried out as follows:

a) Information values are dynamically obtained using 200 queries belonging to the Cranfield 1400th document collection. The information values obtained are then used to reindex a test collection of 25 queries. This query test collection is run as a batch, and retrieval results obtained will therefore be called semi-dynamic.

b) The fully dynamic approach is used in which the 25 queries of the test collection are also run dynamically in the same way as the queries of the update collection.

4. The Results

Before discussing the comparative results of the dynamic versus the static approach, the latter will be examined first. Results obtained earlier with the static approach were not satisfactory [1] and after similar experiments have been done with so-called utility values [4], using the same updating function, the reasons for the failure become clearer. They are the following:

a) The obviously wrong choice of a factor in the updating algorithm which produces increment and decrement steps that are too large.

b) The use of nonoptimally chosen initial (starting) information values. Instead of initializing all information values equal to one (reducing equation (3) to only frequency weights) one could have taken advantage of the available discrimination values for this collection. By using the discrimination values as starting values for the dynamic information values, a better system performance can immediately be obtained. In this case it will be necessary to scale the discrimination values in such a way that the average equals 1 (see c).

c) No provision is made for balancing. Balancing means keeping the average information value equal to 1 in order not to promote or suppress terms not yet updated. In the experiments done with utility values this is shown to be a critical factor.

d) The query test collection was not chosen randomly and after the earlier experiments [1] were done, it was found that the test collection was related to one specific subset of the document collection. This means that keywords specific for that particular field are not likely to be updated by the updating query collection which concerns other topics.

Since the same defects have to be valid for the dynamic strategy, one may not expect an absolute improvement of the system performance, but rather a relative improvement over the statistically obtained information values is anticipated.

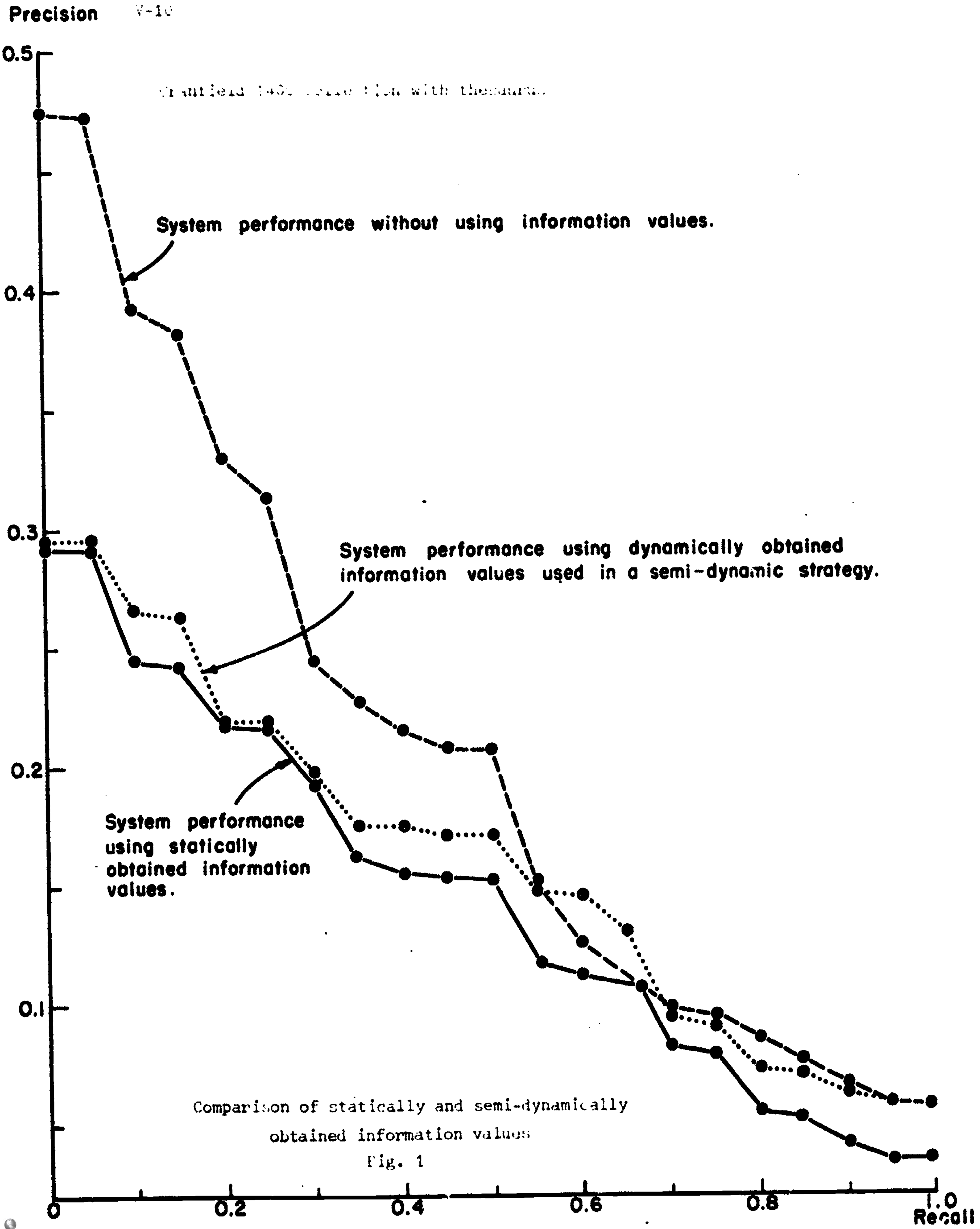
The results of the present experiment A (Fig. 1) indicate that running the test collection in a batch using information values dynamically obtained from the updating collection gives better results than when the static values are used. The improvement is slight though, because the test collection has unfortunately not been chosen randomly. Since the query test collection is directed to a specific field and thus utilizes keywords describing this field, the associated information values are not likely to have been updated by the update collection in which they did not occur.

Much more interesting therefore is experiment B, where the query test collection itself is used in a dynamic way (Fig. 2). Here the concepts of each subsequent query are updated by the previous one.

If the dynamic approach really works as it should, that is, if the good properties of the feedback mechanism mentioned earlier are utilized, this experiment should show clearly the dynamic effects. Fig. 2 shows indeed that system performance is improved considerably compared with both the static and the semi-dynamic case.

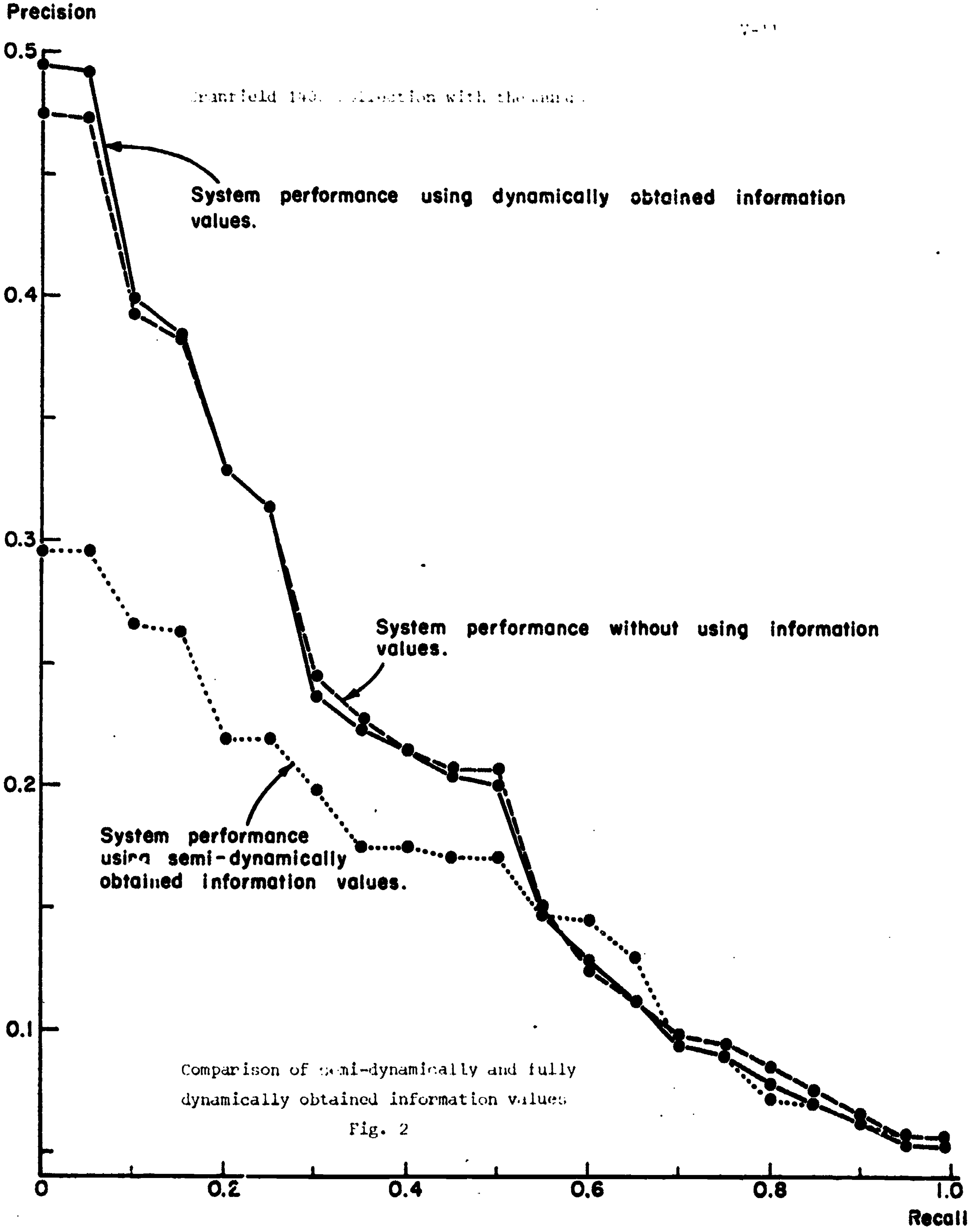
5. Conclusion

For a given collection and a given updating algorithm the static and dynamic updating strategy have been compared. The static information values were obtained in an earlier project and did not perform very well, partly because of defects in the updating algorithm discovered later. In order to draw a fair comparison between statically



Comparison of statically and semi-dynamically obtained information values

Fig. 1



Cranfield 140 collection with the [illegible]

Comparison of semi-dynamically and fully dynamically obtained information values

Fig. 2

and dynamically obtained information values all conditions are kept the same.

The results show that in the two dynamic experiments carried out, dynamic updating is superior to static updating. The first experiment is a mixture of dynamic and static updating in that the testcollect is processed as a batch; the second experiment is fully dynamic and the results of this experiment show in particular a considerable improvement over the static strategy.

References

[1] A. Wong, R. Peck, and A. van der Meulen, An Adaptive Dictionary in a Feedback Environment, Report No. ISR-21, Dept. of Computer Science, Cornell University, December 1972.

[2] M.D. Kerchner, Dynamic Document Processing in Clustered Collections, Report No. ISR-19 to the National Science Foundation, Information Storage and Retrieval, Department of Computer Science, Cornell University, December 1971.

[3] C.R. Sage, R.R. Anderson, and P.F. Fitzwater, Adaptive Information Dissimination, American Documentation, Vol. 15, No. 3, July 1965.

[4] A. Wong and A. van der Meulen, The Use of Utility Values for Dynamic Query-Document Retrieval, Project Report, Department of Computer Science, Cornell University Fall 1972.

Automatic Thesaurus Construction Through The Use of Pre-Defined Relevance Judgments

Kenneth Welles

Abstract

A method for totally automatic construction of thesauri is proposed which relies upon accumulation of match-mis-match influences to converge. Results so far, while not conclusive, are promising.

1. Introduction

It is now an accepted fact that the use of term classification thesauri improve the operation of information storage and retrieval systems. There are many different approaches to the construction of such a thesaurus, ranging from completely by hand to totally automatic. This paper deals with a proposed system of creating a thesaurus totally automatically.

Automatic term classification algorithms have mainly centered on two areas, term co-occurrence as an indication of synonymy, and predetermined relevance judgments as a source of information about synonymy. My work deals with the latter aspect. The basis for such a system is a set of documents, a set of requests, and a set of judgments as to which documents should and should not be retrieved by each request. From this information, the machine can construct a set of classifications of terms (synonym classes) which improve the retrieval effectiveness of the given queries substantially. The rationale behind such a construction is this: if the set of requests was comprehensive, then any similar requests presented in the future will also benefit from the resulting thesaurus.

The starting point in this area is a paper by Jackson of the construction of precisely such a system. [1] A program was written to the specifications of Jackson's paper. Early in the project it was discovered that a minimal collection (about 80% of the ADI collection of documents and queries) requires more than 30 minutes to run on the 360-65 without attaining the first of several iterations. This is obviously an impractical program to implement.

Examination of the program shows that a substantial portion of the run time is spent on constructing, maintaining, and observing what Jackson calls "degeneracy conditions", which assure convergence. A different approach is proposed in this paper. In order to decide which terms are to be considered synonymous, each pair of terms is examined. If these two terms are considered synonymous, then any query-document pair where one term is in the query and the other term is in the document, will have this term pair counted as a match. This increases the number of matching terms in this query-document pair, and so increases the calculated matching coefficient for this query-document pair. If the query-document pair is defined as relevant, then this increase in matching is good. If the query-document pair is defined as not relevant, then this increase is bad (for the purposes of constructing a thesaurus which causes the calculated matches to agree with the defined relevancies). If the amount of good outweights the bad, the pair is considered synonymous, and when all such synonym pairs have been considered, the query-document set is re-examined for the next iteration. Hopefully, after several iterations, the set of synonym pairs will stabilize in a way that gives the calculated relevance of the query-document set closest resemblance to the defined relevance set.

2. Terminology and Definitions

Before continuing with the technical aspects of the proposed program, some definitions are necessary. There are four possible term classes, and for any given query, document and term, the term falls into one of these classes (see figure 1). If the term is not present in either the query or the document, it is class A. If the term is in the query, but not in the document, it is in class B. If the term is in the document but not in the query, it is in class C. Finally, if the term is in both query and document, then the term is in class D.

All correlations between queries and documents are calculated during each program iteration (since the correlations change from one iteration to the next). For each query, the correlation values of the defined relevant and defined not relevant documents are considered. The lowest correlation value of a defined relevant document and the highest correlation value of a defined not relevant document are determined. A value midway between these values is taken as the match value (see figure 2). If any query-document pair exhibits a correlation higher than this match value, then this pair is said to be calculated relevant. If the correlation is less than or equal to this match value, the pair is said to be calculated not relevant. Each query-document pair falls into one of four classes (see figure 3). If the pair is calculated not relevant and defined not relevant (by the initial relevance data), then it is in class R. If the pair is calculated relevant, but defined not relevant, it is in class S. If the pair is calculated not relevant, but defined relevant, it is in class T. If the pair is defined and calculated relevant, it is in class U.

If all the query-document pairs are either class R or class U, then the calculated matches all correspond to the defined matches and no further modifications are needed. However, if there exist query-document pairs in classes S or T, the term space must be modified to cause calculations to agree with definitions.

The cosine correlation is used to calculate the match between a query and document. If the possibility of synonyms is ignored, this value becomes:

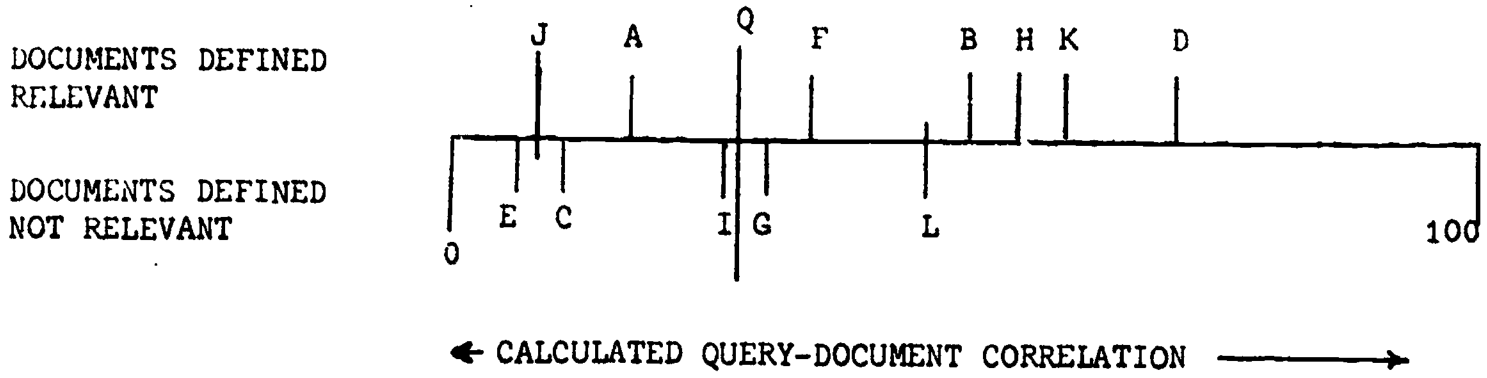
$$\frac{D}{B+C+D}$$

where B, C and D are the number of terms in classes B, C and D respectively. However, if any term in class B is a synonym of any term in class C, then this pair of terms is considered a match, and is counted as a class D term instead of a class B term. All term pairs with one class B term and one class C term are considered "potential synonym pairs" for this

	NOT IN THE DOCUMENT	IN THE DOCUMENT
NOT IN THE QUERY	A	C
IN THE QUERY	B	D

Term Class Definitions

Fig. 1


DOCUMENTS DEFINED RELEVANT
DOCUMENTS DEFINED NOT RELEVANT
J
A
Q
F
B
H
K
D
E
C
I
G
L
0
100
← CALCULATED QUERY-DOCUMENT CORRELATION →

LOWEST DEFINED RELEVANT = J
HIGHEST DEFINED NOT RELEVANT = L
MATCH VALUE = (J+L)/2 = Q

Definition of Match Value

Fig. 2

	CALCULATED NOT RELEVANT	CALCULATED RELEVANT
DEFINED NOT RELEVANT	R	S
DEFINED RELEVANT	T	U

Query-Document Class Definitions

Fig. 3

particular query-document pair. It is seen here that if many of these "potential synonym pairs" are, indeed, considered synonymous, then the correlation between query and document can be raised considerably. It should also be noted that any synonym pair may cause an increase in many different query-document correlations, some desirable, and some undesirable.

As an example, take the query and two documents in table 1(a). In table 1(b) we see that both possible query-document pairs have only one exact term match, and each also has four potential synonym pairs. In table 1(c) we see the effect on the cosine correlation of these query-document pairs if different potential synonym pairs are considered synonymous. If no synonyms are considered, both query-document pairs have the same correlation. However, if RED and GREEN or BALL and BAT are considered synonymous, then the correlation of query-document pair A is raised but that of query-document pair B is not. If BALL and GREEN are considered synonymous, the correlation of both query-document pairs are raised. Also, if BLUE-CHILD is synonymous, then the correlation of query-document pair B is raised, and that of query-document pair A is not.

We can now see that proper choice of synonyms allows a great deal of manipulation of matching coefficients. If, for instance, we had defined query-document pair A as relevant, and query-document pair B as not relevant, then RED-GREEN, or BAT-BALL would be good potential synonym pairs to consider synonymous, while BALL-GREEN or BALL-CHILD would not.

a)

	TERMS CONTAINED		
QUERY	RED	BALL	CHILD
DOCUMENT A	GREEN	BAT	CHILD
DOCUMENT B	RED	GREEN	BLUE

b)

	QUERY-DOCUMENT A	QUERY-DOCUMENT B
TERM MATCHES	CHILD	RED
POTENTIAL SYNONYM PAIRS	RED-BAT RED-GREEN BALL-BAT BALL-GREEN	CHILD-BLUE CHILD-GREEN BALL-BLUE BALL-GREEN

c)

SYNONYMS	COSINE-CORRELATION OF QUERY WITH	
	DOCUMENT A	DOCUMENT B
none	.2	.2
RED-GREEN	.5	.2
BALL-BAT	.5	.2
BALL-GREEN	.5	.5
BLUE-CHILD	.2	.5

Query-Document Calculated Correlations
Under Synonym Rules

Table 1

3. Pseudo-Classification Procedure

A term matrix is set up which has a numerical value for each term pair. If this value exceeds a user-defined "threshold of synonymy", then the corresponding pair of terms is considered synonymous (for use in calculating the matching function). The term matrix is initially set to all zeroes (no synonyms).

At the start of each iteration, all query-document cosine correlations are calculated and stored. The correlations are calculated not only with direct term matches, but also counting any pair of terms which is a "potential synonym pair" for this query and document, and which is defined synonymous by the term matrix. Since the entries in the term matrix vary from one iteration to the next, the calculated correlations will also vary.

After all the correlations have been stored, each query-document pair is again considered in turn, and the previously stored correlation value and the query-document class (R, S, T or U) to which this pair corresponds is used to calculate a number. This number, which may be positive or negative, is the "term modifier" and is added to each entry in the term matrix which corresponds to a "potential synonym pair" for this particular query-document pair.

After this calculation, the number of query-document pairs in classes S and T is counted and output as an indication of the degree of convergence to the desired state. The modified term matrix is then utilized in the next iteration for calculation of the matching function. Iteration continues until the program converges (all query-document pairs are in class R or U) or operator intervention occurs.

The heart of the program is the action of the term modifier. Any one term pair may be a "potential synonym pair" for many different query-document pairs. Thus, after each iteration, the corresponding term matrix entry will be changed by an amount equal to the sum of the term modifiers for all query-document pairs which include this term pair as a "potential synonym pair."

If this set of query-document pairs consists entirely of class T pairs, then one would wish to raise the correlation of the pairs so that they might become class U pairs. If the modifier is a positive number for a class T query-document pair, then the net effect of many class T pairs will be to raise the term matrix entry above the threshold of synonymy. This, in turn, would increase the correlation of the query-document pairs, which is the desired result.

Conversely, if we are presented with class S query-document pairs, we wish to lower the correlation by causing synonym pairs which contribute to the correlation to become "potential synonym pairs" below the "threshold of synonymy," i.e., to cause them to no longer be synonymous. If the term modifier is negative for class S query-document pairs, it will have the desired effect.

Query-document pairs of the S or T classes are called mismatched pairs because the calculated results do not agree with the defined relevancies. The degree of mismatch is the amount that the calculated correlation differs from the match value. If the degree of mismatch is large, then many "potential synonym pairs" which are (or are not) synonyms must be modified until they are not (or are) synonyms. If a term modifier is made large in magnitude, then it will have a greater effect on the term matrix than other smaller term modifiers. This causes (on the average) a greater amount of change in the number of "potential synonym pairs" which actually change status to or from synonymy. It is therefore desirable that the term modifier should vary in magnitude with the degree of mismatch in classes S and T.

At first it would seem that, because query-document pairs in classes R and U need not be changed, the corresponding term modifiers should be zero. This was tried and found to cause oscillation and prevent convergence. The reason is that the correctly matched pairs do nothing to maintain their status quo. When the mismatched query-document pairs modify the term space to correct their own mismatch, they disturb the balance of synonymy in correctly matched pairs. To prevent the resultant oscillation of query-document pairs between classes R and S, and classes T and U, it is necessary to give term modifiers of classes R and U small negative and positive values respectively. The term modifier of class R should be proportional to minus the number of pairs in class T and U. The term modifier of class U should be proportional to the number of pairs in classes R and S. This assures stability of solution in the fully or nearly fully convergent case (almost all pairs in class R or U).

To assure continuity of the term modifier in class S, the term modifier is equal to the term modifier of class R (a constant) minus the absolute value of the mismatch of the class S query-document pair. Similarly, the class T term modifier is equal to the value of the class U term modifier plus the absolute value of the mismatch.

4. Programming System

A sample program as implemented in FORTRAN IV is shown in the appendix, as an example of this algorithm. All arithmetic is performed with integer and binary variables and arrays. Binary variables are treated as integers with value 0 or 1.

In section A, the document vectors, query vectors, and relevance judgments are read in. This is the main data. The term-term matrix and synonym matrix are zeroed, and initial values for all other variables are set up.

In section B, the correlation coefficient is calculated, taking into account (statements 500 and on) any synonym term matches as well as direct term matches. The calculated correlation of query (J) and document (I) is stored in correlation (J, I).

In section C, the dynamic matching threshold is calculated for each query, and stored in matchvalue (J).

In section D, each query-document pair is considered in turn. The "term-modifier" (variable name is MODIFIER) is calculated from the relevance judgment (relevant (J, I)) and calculated correlation (correlation (J, I)) of the pair, and the dynamic match value (matchvalue (J)) of this query.

In section E, all term pairs are examined, and all which are "potential synonym pairs" for this query document pair (termterm (K1,K2)) are modified by the "term modifier".

After sections D and E have been completed for all query-document pairs, the modified term-term matrix (termterm (K1,K2) is examined in section F. From the data in this matrix, the synonym

```
C*****THIS PROGRAM CONSTRUCTS A THESAURUS OF TERMS AUTOMATICALLY.
C*****A SET OF "L" DIFFERENT DOCUMENTS AND "M" DIFFERENT QUERIES
C*****ARE USED AS DATA.  EACH DOCUMENT OR QUERY MAY HAVE
C*****ANY OR ALL OF "N" DIFFERENT TERMS PRESENT (BINARY WEIGHTING).
C*****DATA FOR DECISIONS IS GIVEN IN THE BINARY MATRIX OF
C*****DIMENSION (L*N) CALLED DOCUMENT; THE BINARY MATRIX OF
C*****DIMENSION (M*N) CALLED QUERY; AND THE BINARY MATRIX OF
C*****DIMENSION (M*L) CALLED RELEVANT.  THIS LAST MATRIX IS THE
C*****OPERATORS DEFINITION OF WHAT DOCUMENTS ARE RELEVANT TO WHICH
C*****QUERIES.  DEFINED RELEVANCE IS INDICATED BY A "1", AND DEFINED
C*****IRRELEVANCE IS INDICATED BY A "0".

        IMPLICIT INTEGER(A-Z)

        BINARY DOCUMENT(L,N),QUERY(M,N),RELEVANT(M,L)
        BINARY SYNONYM(N,N)
        INTEGER TERMTERM(N,N),CORRELATION(M,L),MATCHVALUE(M)

C******************** SECTION A ********************************

C*****INPUT DATA TO BASE THE THESAURUS ON

        READ  ((DOCUMENT(I,K),K=1,N),I=1,L)
        READ  ((QUERY(J,K),K=1,N),J=1,M)
        READ  ((RELEVANT(J,I),I=1,L),J=1,M)

C*****ZERO OUT THE SYNONYM AND TERM-TERM MATRICES
        DO 100 K1=1,N
        DO 100 K2=1,N
        TERMTERM(K1,K2)=0
        SYNONYM(K1,K2)=0
100     CONTINUE

        ITERATION=0
        MAXITERATION=20
        A=5
        B=5
        SYNTHRESHOLD=20
        CRITERION=L*M/20

C******************** SECTION B ********************************

C*****FOR EACH QUERY-DOCUMENT PAIR, CALCULATE THE CORRELATION
C*****COEFFICIENT AND STORE IT
200     DO 1000 J=1,M
        DO 1000 I=1,L
        MATCH=0
        LENGTH=0
        DO 900 K1=1,N
        CONDITION=1+DOCUMENT(I,K1)+2*QUERY(J,K1)
        GO TO (400,500,700,300),CONDITION
```

BEST COPY AVAILABLE

```
C*****TERM IS NOT IN QUERY OR DOCUMENT
400     GO TO 900

C*****TERM IS IN BOTH QUERY AND DOCUMENT
300     LENGTH=LENGTH+1
        MATCH=MATCH+1
        GO TO 900

C*****WHEN CALCULATING THE CORRELATION, TAKE INTO ACCOUNT ALL
C*****THE SYNONYMS WHICH ARE INDIRECT TERM MATCHES.
C*****TERM IS IN DOCUMENT, BUT NOT IN QUERY
500     LENGTH=LENGTH+1
        DO 600 K2=1,N
        IF (QUERY(J,K2) .EQ. 0) GO TO 600
        IF (SYNONYM(K1,K2) .EQ. 0) GO TO 600
        IF (DOCUMENT(I,K2) .EQ. 1) GO TO 600
        MATCH=MATCH+1
        GO TO 900

600     CONTINUE
        GO TO 900

C*****TERM IS IN QUERY, BUT NOT IN DOCUMENT
700     LENGTH=LENGTH+1
        GO TO 900

900     CONTINUE
C*****NORMALIZE THE CORRELATIONS TO A 0-100 SCALE
        CORRELATION(J,I)=(MATCH*100)/LENGTH
1000    CONTINUE
C*****ALL CORRELATIONS OF Q-D PAIRS HAVE BEEN CALCULATED AT THIS POINT

C********************SECTION C ******************************

C*****FOR EACH QUERY....

        DO 1300 J=1,M
        HIGHIRRELEVENT=0
        LOWRELEVANT=100

        DO 1200 I=1,L
        IF (RELEVANT(J,I) .EQ. 1) GO TO 1100

C*****FIND THE DEFINED IRRELEVENT DOCUMENT WITH THE
C*****HIGHEST CORRELATION COEFFICIENT.
        HIGHIRRELEVENT=MAX(HIGHIRRELEVENT,CORRELATION(J,I))
        GO TO 1200

C*****AND FIND THE DEFINED RELEVANT DOCUMENT WITH
C*****THE LOWEST CORRELATION COEFFICIENT.
1100    LOWRELEVANT=MIN(LOWRELEVANT,CORRELATION(J,I))
1200    CONTINUE

C*****AND SET THE MATCH VALUE LEVEL MIDWAY BETWEEN THESE VALUES.
        MATCHVALUE(J)=(LOWRELEVANT+HIGHIRRELEVENT)/2
1300    CONTINUE

C*********************SECTION D ******************************
```

```
      RCASES=0
      SCASES=0
      TCASES=0
      UCASES=0

      DO 2000 J=1,M
      DO 2000 I=1,L

      IF (CORRELATION(J,I) .GE. MATCHVALUE(J)) GO TO 1500
      IF (RELEVANT(J,I) .EQ. 1) GO TO 1400

C*****THIS DOCUMENT IS CALCULATED IRRELEVANT, AND
C*****DEFINED IRRELEVANT, IT IS A CASE "R"
      MODIFIER=-A
      RCASES=RCASES+1
      GO TO 1700

C*****THIS DOCUMENT IS CALCULATED IRRELEVANT, AND
C*****DEFINED RELEVANT, IT IS A CASE "T"
1400  MODIFIER=+ABS(CORRELATION(J,I)-MATCHVALUE(J))+B
      TCASES=TCASES+1
      GO TO 1700

1500  IF (RELEVANT(J,I) .EQ. 1) GO TO 1600

C*****THIS DOCUMENT IS CALCULATED RELEVANT, AND
C*****DEFINED IRRELEVANT, IT IS A CASE "S"
      MODIFIER=-ABS(CORRELATION(J,I)-MATCHVALUE(J))-A
      SCASES=SCASES+1
      GO TO 1700

C*****THIS DOCUMENT IS CALCULATED RELEVANT, AND
C*****DEFINED RELEVANT, IT IS A CASE "U"
1600  MODIFIER=+B
      UCASES=UCASES+1

1700  CONTINUE

C**********************SECTION E *********************************

C*****AT THIS POINT, THE MODIFIER HAS BEEN DEFINED ACCORDING TO
C*****WHAT CASE THE QUERY DOCUMENT PAIR IS AT THIS ITERATION.

      DO 1900 K1=1,N
      IF (DOCUMENT(I,K1) .GE. QUERY(J,K1)) GO TO 1900
      DO 1800 K2=1,N
      IF (DOCUMENT(I,K2) .LE. QUERY(J,K2)) GO TO 1800
C*****AT THIS POINT, TERM K1, AND TERM K2 ARE A POTENTIAL
C*****SYNONYM PAIR FOR THIS QUERY-DOCUMENT PAIR,
C*****SO THE CORRESPONDING ENTRY IS MODIFIED.

      TERMTERM(K1,K2)=TERMTERM(K1,K2)+MODIFIER
      TERMTERM(K2,K1)=TERMTERM(K1,K2)
1800  CONTINUE
1900  CONTINUE
2000  CONTINUE
```

```
C******************SECTION F ********************************

C*****NOW THE SYNONYM MATRIX IS REDEFINED BY
C*****WHICH TERM-TERM ENTRIES ARE NOW ABOVE THE THRESHOLD
C*****OF SYNONYMY.
        DO 2200 K1=1,N
        DO 2200 K2=1,N
        IF (TERMTERM(K1,K2) .GT. SYNTHRESHOLD) GO TO 2100
        SYNONYM(K1,K2)=0
        GO TO 2200

2100    SYNONYM(K1,K2)=1
2200    CONTINUE

C********************SECTION G ******************************

C*****FIND OUT IF THE PROGRAM HAS COVERGED OR EXCEEDED
C*****THE MAXIMUM NUMBER OF ITERATIONS.  IF NOT, LOOP!
        ITERATION=ITERATION+1
        WRITE  ITERATION,RCASES,SCASES,TCASES,UCASES
        TOTALMISMATCH=SCASES+TCASES
        IF (TOTALMISMATCH .LE. CRITERION) GO TO 2300
        IF (ITERATION .LE. MAXITERATION) GO TO 200
2300    WRITE  ((SYNONYM(N1,N2),N1=1,N),N2=1,N)
        STOP
        END
```

matrix is changed to reflect the updated state of calculated synonymy.

In section G, program status is printed out and a decision is made whether to iterate (back to section B) or print out and halt.

5. Implementation

A program essentially identical to this, but modified for a DEC PDP-11/20 was run for small and large collections of data. [illegible] small data sets, convergence was reached in two to five iterations. The larger data set was about 80% of the ADI collection, and while results are promising, convergence on this data set has not yet been attained.

References

[1] D.M. Jackson, The Construction of Retrieval Environments and Psuedo-Claaification Based on External Relevance, Information Storage and Retrieval, Vol. 6, p. 187-219, 1970.

Content Analysis and Relevance Feedback

A. Wong, R. Peck, and A. van der Meulen

Abstract

Content analysis is a vital part of any automatic document retrieval system where natural language has to be analyzed in order to detect the information carrying parts. The assignment of appropriate identifiers to the documents and queries — "the indexing process" — can be carried out on different levels of complexity which generally agree with different levels of system performance.

A device for indexing improvement of a quite different nature is the "user feedback" technique which can be applied in an interactive retrieval system. The initial indexing which is a result of a rather imperfect content analysis can be corrected and improved by using the judgment of the user concerning the relevancy or non-relevancy of his retrieved documents.

This report deals with the question: how critical is the quality of the content (language) analysis which results in the initial indexing in an interactive retrieval system? Since in such a system feedback techniques will improve system performance substantially one could doubt if original defferences in content analysis will still affect the final performance. If such differences in indexing refinement turn out to be retained after the feedback is applied, every improvement in initial indexing should be put into practice; in addition a good justification exists for working in the area of automatic dictionary construction.

1. Introduction

The performance of a document retrieval system depends heavily on the transformation of the natural language of the document into an artificial retrieval language. The document description in such a retrieval language results in a much shorter representation compared with the original one, and it is this indexing process which determines how effectively the document can be retrieved.

If one defines as "ideal indexing process" as a process which results in retrieval of only relevant documents in response to queries, it is likely that, even with the most refined techniques, ideal indexing does not exist. The two reasons are:

- the transformation of the author's ideas into a written text may be imperfect; moreover, during indexing usually only title and abstract are considered.
- the existing language analysis tools are imperfect.

The first reason in particular will always limit the quality of the indexing process even if ideal language analysis devices were available.

When dealing with large collections of documents, the automation of text analysis becomes a necessity, since manual indexing may not then be a realistic alternative. Evaluation tests for a long period have shown that manual indexing was superior to automatic indexing. Nowadays the roles incline to change. This is mainly due to the fact that in modern interactive retrieval systems the implementation of user feedback strategies (index corrective mechanisms) yield a considerable improvement in system performance. To be sure, those interactive strategies are in principle also applicable to computerized retrieval systems supplied with manually indexed documents; but usually such a system organization does not allow the implementation of an effective feedback algorithm.

Concentrating first on the automatic content analysis tools, one may distinguish two basically different approaches:

a) Non-lingusitic computer techniques.

Examples are:

- the automatic stemming of words followed by the counting of their frequency of occurrence;
- procedures for the automatic detection of common words;
- statistical procedures for the automatic construction of dictionaries; and
- the automatic creation of weighted dictionaries, where the weight reflects the description power of an identifier.

b) Computer oriented linguistic techniques.

In those procedures the syntactical meanings of sentences are taken into account rather than mere keywords; also phrases may be recognized in an explicit way.

Such a syntactical analysis, was supposed to fill the gap between a pure mechanical analysis (category a) and an intellectual manual one. To date the situation however is such that application of the now existing linguistic techniques deteriorate system performance, rather than providing a substantial improvement. In this report therefore only statistical language analysis will be considered (category a).

The retrieval results obtained with the available automatic indexing devices are far from satisfactory, and one might doubt if it ever will become possible to improve the initial indexing sufficiently.

To simplify the indexing problem, user feedback techniques may be used, based on the premise that index corrections can be made dynamically during the course of the search by utilyzing the judgment of the user concerning the relevancy of his retrieved documents [1]. One may refer to "relevance feedback" and "document modification" as corrective indexing techniques which allow both the reindexing of a query (relevance feedback) as well the reindexing of documents (document modification). It should be mentioned here that those corrected indexes are not static entities, but dynamic, in tune with actions performed by the system users.

Relevance feedback in particular has proved to be a powerful technique which increases system performance significantly, since queries are in general short and poorly formulated. One must ask then how imperfect the initial indexing may be while still yielding results after feedback application which are comparable to those of the more refined initial indexing; in other words is the final feedback result a function of the initial indexing?

If final results appear to be independent of initial indexing, most efforts concerning content analysis and in particular dictionary construction might be less meaningful than they originally appear to be. If those results show some improvement due to better initial indexing what compromise must be made between indexing expedience and returns in terms of system performance after feedback is applied? But if the final results are highly dependent on the quality of initial indexing the answer is clear. All efforts directed towards content analysis improvement remain of particular importance.

2. Experiments

The purpose of the experiments is to investigate the influence of language analysis tools on final system performance after a sufficient number of feedback iterations have been executed. "Sufficient" here means that a new iteration will not further affect the obtained system performance. In the SMART environment two or at most three iterations are normally sufficient. The experiments are likely to be dependent on the type of feedback algorithm used, which will be Rocchio's in all cases.

A) Used Language Analysis Tools

The main analysis tools provided by SMART include three dictionaries:

- the word-form dictionary,
- the word-stem dictionary, and
- the thesaurus

The word-stem dictionary (suffic deletion) is a refinement of the word-form dictionary (plural s endings deletion), and the thesaurus (grouping of related terms) is a refinement of the word-stem dictionary.

An improvement which can be applied to each of the existing dictionaries is the application of the so-called "discrimination values" [2], that is, of quantities which reflect the descriptive power of the dictionary items (terms). More specifically, the discrimination value is a measure of the change in average correlation of a document collection to its center-of-mass (centroid), measured first using the term as an index, and again with that term deleted. If the collection moves closer together, when term i is deleted, that is, if the average correlation with the centroid increases, that term is valuable in distinguishing individual documents.

The application of discrimination values can be carried out in two possible ways:

- deletion of bad discriminators which process however will not be considered in this report, and
- creation of a weighted thesaurus [3].

B) Comparisons

Two collections are considered in order to justify generalizing the results obtained in these experiments. They are:

- the TIME collection consisting of 425 documents in the political science field provided with 83 queries, and
- the ADI collection consisting of 82 documents in documentation provided with 32 queries.

Two main experiments are carried out: first a comparison is made between four different dictionaries using the TIME collection. Compared are the system performances using a word-form dictionary and a thesaurus both with and without discrimination value application. System performances before and after two feedback iterations are considered. Second a comparison of a word-stem dictionary and a thesaurus using the ADI collection is made. System performances before and after three feedback iterations are considered.

In the comparison of feedback results no attempt is made to go into complex evaluation schemes [4,5]. The system performances are expressed in simple recall-precision curves, which are suitable for the outlined purposes.

3. Experimental Results

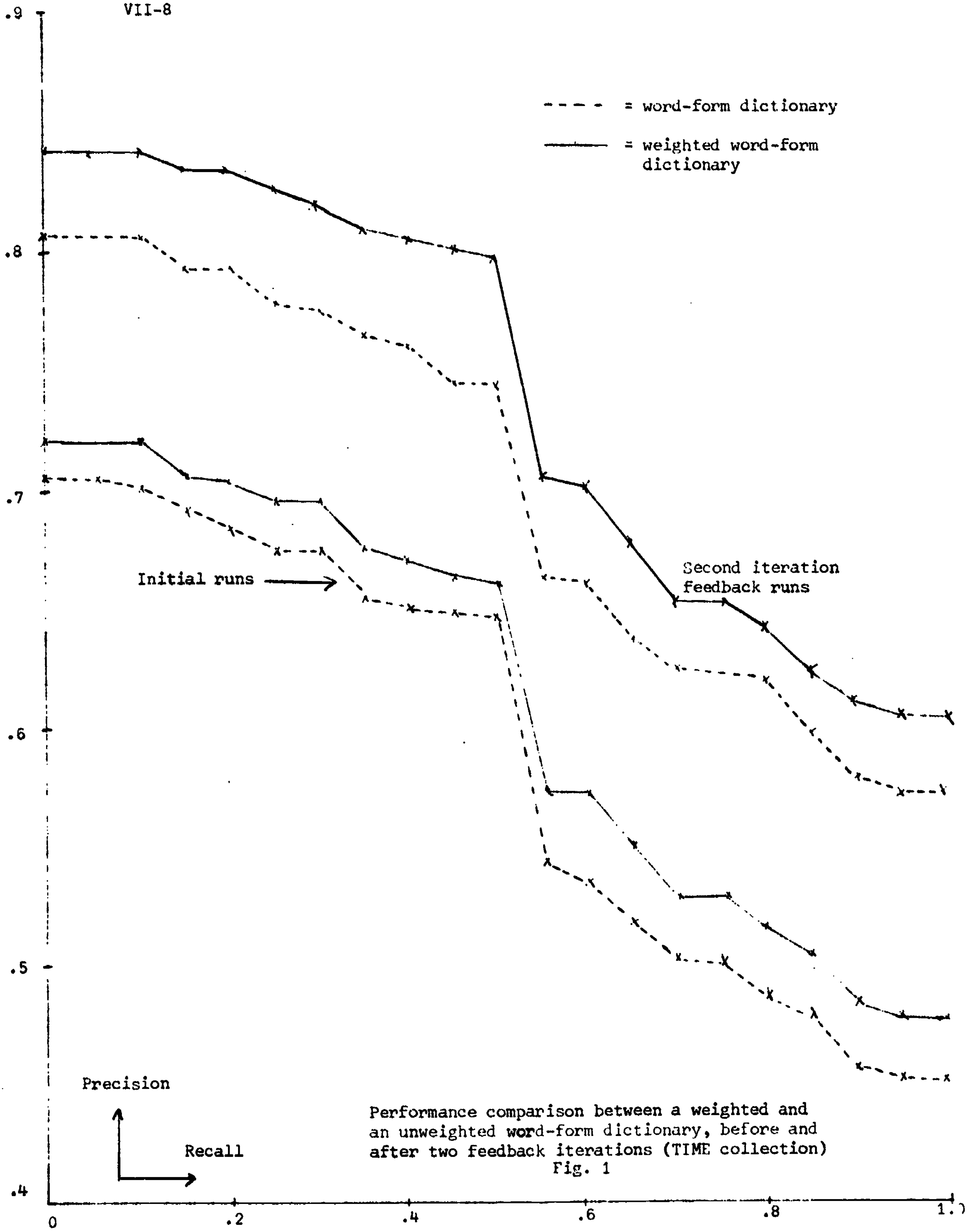
The results for both collections (Figs. 1, 2, 3, and 4) clearly demonstrate that in all the investigated cases differences in initial performance are retained in the final precision-recall curves. In Figures 1, 2, and 3, the word-form dictionary serves as reference curve and is compared with the weighted word-form dictionary, the thesaurus, and the weighted thesaurus. In Fig. 4 a word-stem dictionary is compared with a thesaurus.

From the recall-precision curves one may draw the remarkable conclusion that the shape of each curve, reflecting a special dictionary performance, remains invariant after the feedback operations, however the position of the curves is lifted. Also the relative ordering of results of various dictionaries remains invariant.

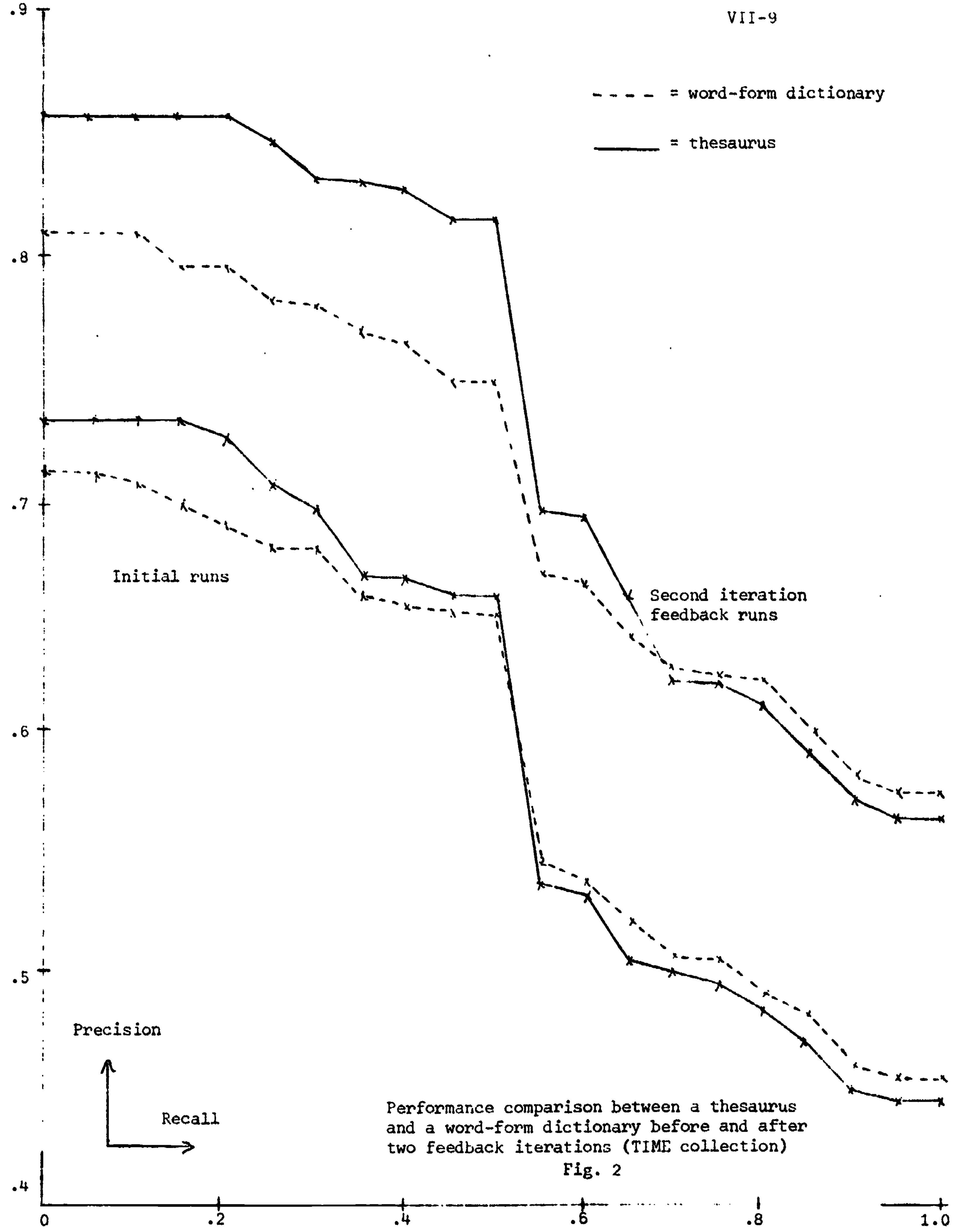
In the case of the TIME collection a better initial performance curve is inclined to lift relatively more (Figs. 1, 2, and 3); this results in a spread out of the final curves. The ADI collection shows a slight "wash-out" effect in that initial differences are diminished. It has to be noted that the initial word-stem dictionary performance is fairly poor (low recall-precision curve), which explains the wash-out effect.

4. Conclusion

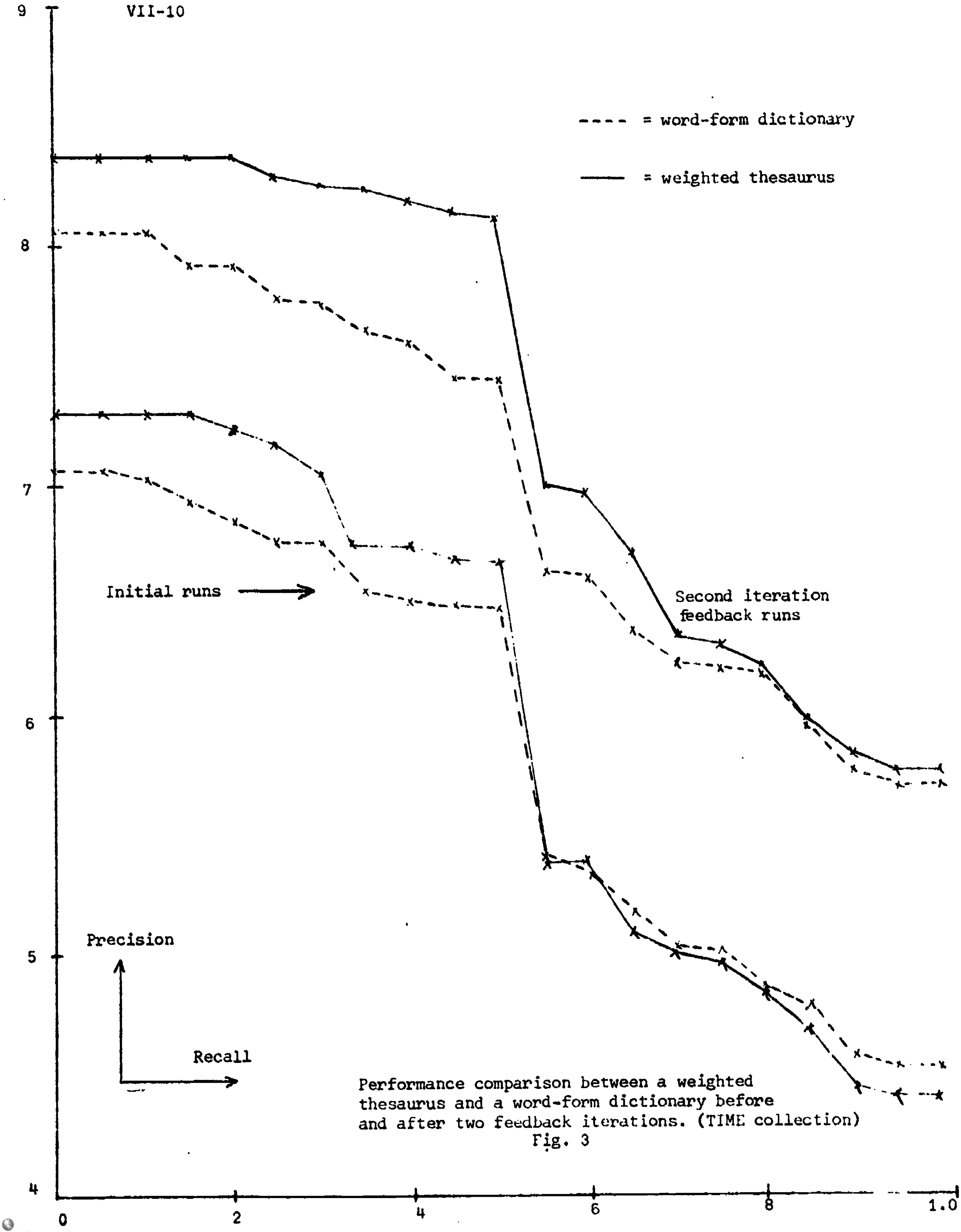
The results indicate clearly that the final system performance, that is, the final retrieval result after user feedback is applied, is highly dependent on the system performance of the initial indexing process.



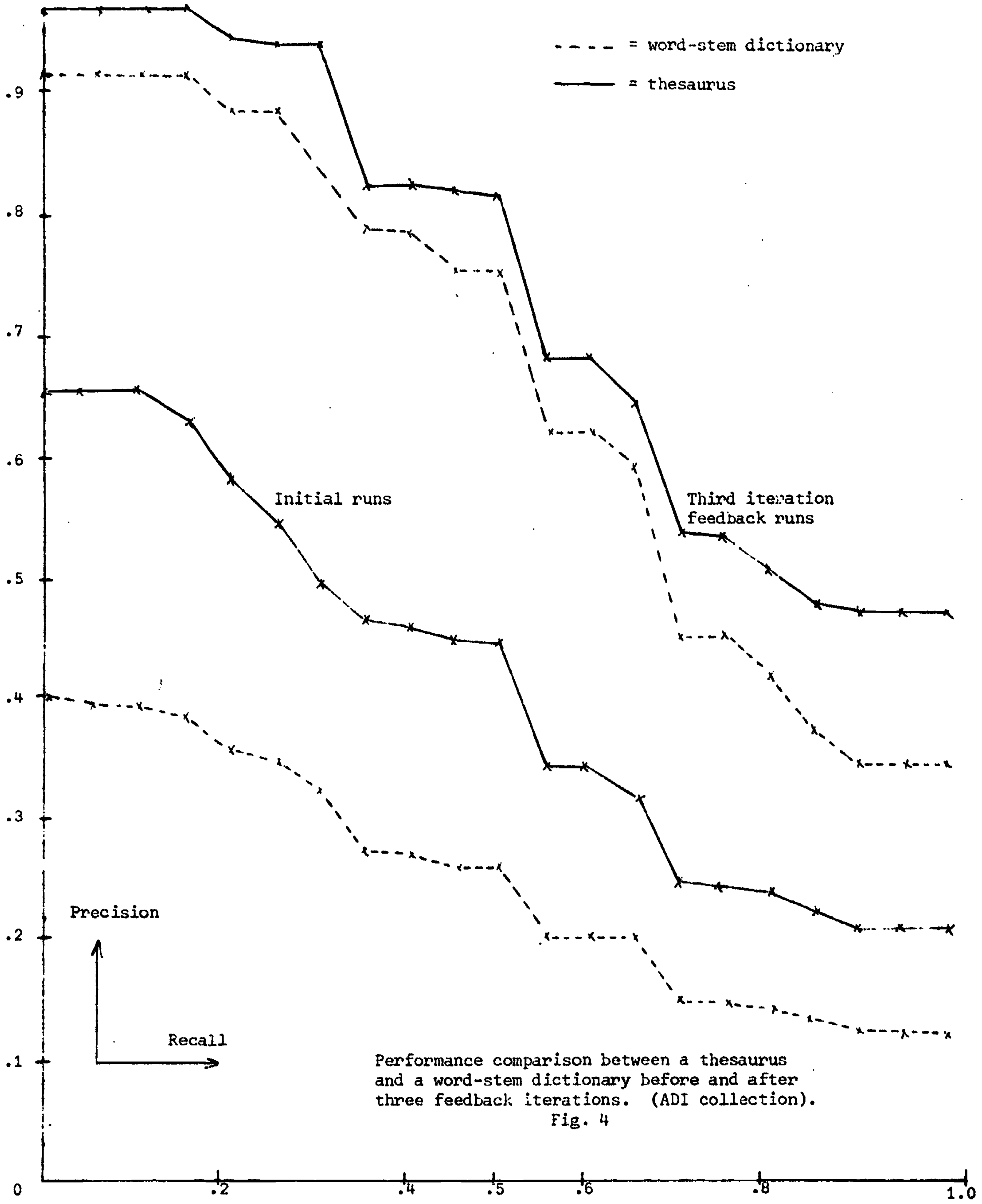
Performance comparison between a weighted and an unweighted word-form dictionary, before and after two feedback iterations (TIME collection)
Fig. 1



Performance comparison between a thesaurus and a word-form dictionary before and after two feedback iterations (TIME collection)
Fig. 2



Performance comparison between a weighted thesaurus and a word-form dictionary before and after two feedback iterations. (TIME collection)
Fig. 3



Performance comparison between a thesaurus and a word-stem dictionary before and after three feedback iterations. (ADI collection).
Fig. 4

It must be noted that this conclusion is derived for the application of Rocchio's feedback algorithm; other feedback mechanisms such as for example the replacement of the original query by the index of a retrieved relevant document, might yield different results. Unfortunately no evaluations between different feedback strategies are available, but Rocchio's is at least the most established one.

It is for this feedback strategy that one may state that every tool which improves the indexing performance as an outcome of the content analysis of natural language is beneficial because initial differences in system performance are retained after user feedback is applied.

References

[1] G. Salton, The SMART Retrieval System, Prentice Hall Inc., Englewood Cliffs, N.J., 1971.

[2] K. Bonwit and J. Aste-Tonsmann, Negative Dictionaries, Information Storage and Retrieval Report No. ISR-18, to the National Science Foundation, Section VI, Department of Computer Science, Cornell University, October 1970.

[3] A. Wong, R. Peck, and A. van der Meulen, An Adaptive Dictionary in a Feedback Environment, Term project report, Department of Computer Science, Cornell University, May 1972.

[4] G. Salton, The Performance of Interactive Information Retrieval, Information Processing Letters 1 (1971), North Holland Publishing Company.

[5] C. Cirillo, Y.K. Chang, and J. Razon, Evaluation of Feedback Retrieval Using Modified Freezing, Residual collection and Test and Control Group, Information Storage and Retreival, Report No. ISR-16, to the National Science Foundation, Section X, Department of Computer Science, Cornell University, September 1969.

On Controlling the Length of the Feedback Query Vectors

Karamvir Sardana

Abstract

Various strategies for reducing the lengths of feedback vectors, which get elongated during regular feedback processing, are tested. The results show that the strategies based on the knowledge of the discrimination values of the concepts, are quite successful. The best adjudged method retains the top best discriminating concepts in every feedback query vector.

1. Introduction

A) Indexing

All automatic document retrieval systems use some method or other for converting a natural language document or a query into a form that is representative of the corresponding document or the query and can be stored internally in a computer. This process is known as indexing and is quite important because much of the efficiency of document retrieval systems depends on it. In the SMART automatic retrieval system [1], indexing transforms a piece of natural language text into its representative concept-weight (c-w) vector form meant for internal storage in the machine. A concept or a term is an atomic entity, a word or a phrase used to describe the document whereas the associated weight denotes the concept's importance in the document. In the existing SMART system, which is taken as a test environment in the present study, the weight of a concept is normally a linear function of its frequency of occurrence in the text of a document or a query. In this report, the c-w vectors used in the SMART system are referred to as standard vectors or simply as vectors.

B) Length of a Vector and Importance of Controlling it

The length of a document or a query vector (referred to simply as a vector in the sequel) is defined to be the number of index terms or concepts (that is, the ones with nonzero weights) constituting the vector. The length of a vector affects the overall retrieval process in the following ways:

i) The storage space occupied by a vector depends on the number of c-w pairs, that is, the length of the vector, and the storage needed for one c-w pair. In the present SMART system, each c-w pair occupies one computer word of storage.

ii) The process of correlating two vectors is frequently used for document searching in retrieval systems. The correlation measure widely used in the SMART system can be graphically interpreted as the cosine of the angle between the two vectors in the n-dimensional Euclidean space. The cosine correlation coefficient between the vectors

$P = (p^1, p^2, \ldots, p^n)$, where p^i is the weight of the i^{th} concept

and

$Q = (q^1, q^2, \ldots, q^n)$

is computed as

$$\mathrm{COR}(P,Q) = \frac{\sum_{i=1}^{n} p^i q^i}{[\sum_{i=1}^{n} (p^i)^2]^{1/2} [\sum_{i=1}^{n} (q^i)^2]^{1/2}} .$$

When very long query vectors are matched with the document vectors, the cosine coefficient tends to be small because a factor proportional to the vector magnitude* appears in the denominator [7]. If the query vector is reasonably short, the vector magnitude is smaller which implies a larger value for the correlation coefficients. If the user has specified a threshold value in the correlation coefficient to distinguish retrieved from nonretrieved documents, the use of a shorter query vector will produce a larger number of output documents than the corresponding longer query vector.

iii) The time required for the correlation process, using an algorithm that stores only the nonzero weight concepts of the vectors (as is done in the SMART system), depends on the length of each vector. This is so because one must compare the two lists of concept numbers in order to determine the matching concepts.

Thus, it is important that during various phases of processing through the retrieval system, such vectors not only be

*As used here, the vector magnitude of a vector P is $[\sum_{i=1}^{n} (p^i)^2]^{1/2}$ and the length of a vector is the number of concepts with nonzero weights in the vector. These definitions of vector magnitude and the length of a vector are consistent with Murray's [2] definitions. Note that Salton's [7] definition of vector length is different from the corresponding definition used here and is the same as that of the vector magnitude in the present context.

fully representative of their intended meaning but also be short and concise to give high correlation coefficients and to save on storage and searching costs.

This implies that there is a definite need for controlling the length of vectors which have a tendency to grow long as a result of retrieval processing. This must be done by trimming the elongated and unwieldy vectors such that their shorter versions carry the gist of the correspording originally long vectors.

2. Earlier Results

A) Murray's Strategy for Reducing the Vector Lengths

Earlier experiments in this area have been conducted by Murray [2] for controlling the lengths of profile vectors. A profile vector (cluster centroid) is a vector that represents all the vectors in a cluster. Murray suggests reducing the lengths of profile vectors by chopping off 80% or so of the concepts with the lowest weights (and thus the frequencies in the standard vectors), which according to his recommendations, results in only a slight decrease in retrieval performance. This method can similarly be used for reducing any vector length for that matter.

The idea is to remove those concepts in a vector which because of their relatively small weights, will not affect the orientation of the vector in the vector space by much and, therefore, will not appreciably influence the correlations of this vector with any other. Further, a detailed analysis by Murray shows the following justification for using this strategy:

Let P be a vector whose length is to be reduced and Q be another vector with which the vector Q is to be (cosine) correlated during the course of document searching. Then, the contribution to the total correlation, by matching-concepts with weights p^i and q^i, is

$$\text{CONTRIBUTION} = \frac{p^i}{|P|} \cdot \frac{q^i}{|Q|} = \frac{p^i}{[\sum_{j=1}^{n} (p^j)^2]^{1/2}} \cdot \frac{q^i}{[\sum_{j=1}^{n} (q^j)^2]^{1/2}} .$$

For a fixed vector Q, the values of $q^i/|Q|$ are fixed and variations in contribution are due to $p^i/|P|$, called the correlation contribution ratio.

Now, as in the present strategy for reducing the vector lengths, the lowest weight terms of vector P are thrown out, the correlation loss due to these terms is small and the retrieval performance is not affected.

This strategy is valuable because a vector can be trimmed to only 20% of its original length while sacrificing only a little in retrieval performance.

B) Other Related Results by Murray

Some other related and interesting results regarding profile vectors by Murray are presented, to be used later on in the discussion. These results are:

i) Weighted profile vectors are significantly better in performance than unweighted profile vectors.

ii) Profile vectors consisting of concepts whose weights are rank values* give superior performance when base values are small. A rank value is the difference between a base value and the rank assigned to the term if all terms in the vector are ordered by decreasing frequency. The base value is a constant chosen large enough to insure that all weights are positive in the profile vectors.

iii) Profiles with concept weights based on frequency ranks** give performance better than the standard or rank value vectors. Such profiles avoid correlation domination by larger weight terms while at the same time allowing smaller weight terms to have a relatively little more say in determining the correlations.

iv) Selection of good index terms is more valuable than making fine frequency distinctions among important index terms.

v) Using a few broad categories, typically four, of weight classes gives performance equivalent to that obtained by using a larger number of weight classes as used in the standard weighted vectors.

3. Present Problem

A) Origination

Rocchio-type formulae are widely used for relevance feedback, and they have been shown to result in an improved retrieval performance [1]. One side effect of using relevance feedback in this manner is the growth in the length

*The vectors formed in this way are called rank value vectors.

**The frequency ranked vectors are constructed by arranging the concepts constituting a vector in increasing frequency order and then assigning the weight of a concept equal to its rank in such a list. The concepts with the same frequencies are assigned the same rank. It should be noticed that the frequency ranked vectors are essentially the rank value vectors for which the base value is chosen in such a way that the minimum weight of the concepts equals unity.

of the feedback query vectors (fqv's). The elongation of the fqv's can amount to as much as twelve times the average length of the corresponding original query vectors. To some extent, the growth in the lengths of the query vectors is quite desirable because as Murray [2] has also observed, the original queries tend to be short and omit the background material that might really be helpful. On the other hand, too long a set of query vectors with possibly a lot of unimportant terms could damage the retrieval performance in addition to using up more storage and costing more in retrieval searches. The idea, then, is to reduce the elongated fqv's to their optimum length somehow.

Furthermore, it would be interesting to see if relevance feedback can, in some way, be reinforced by using the knowledge of discrimination values (dv's) of concepts in the fqv's. The theory of discriminating power of individual concepts has been expounded by Bonwit and Aste-Tonsman [3] and developed further by Crawford [4].

B) Exact Definition and Scope of the Problem

The problem at hand is to discover strategies to trim the fqv's to vectors of manageable shorter lengths such that the retrieval performance obtained by using the trimmed versions of the fqv's is better than or at least equivalent (if possible) to that obtained by using the original elongated fqv's. Second part of the problem is to discover means to augment the relevance feedback by using the dv's of concepts utilizing ideas of Bonwit and

Aste-Tonsman [3] and Crawford [4]. The results are to be compared with Murray's work [2].

It is worth recalling that some work in feedback reinforcement by term dv's has already been done by Bjorklof [5] and Doeppner, Finley and Peterson [6] using some strategies which have not met with much success.

C) Methods and Solutions in Brief

It is proposed to use the information of dv's of concepts in achieving both the ends. Briefly, the procedure to be used is as follows:

i) Take a document collection and the associated queries. After the original iteration of document searching, do one iteration of regular Rocchio-type feedback. Note its performance and save the resulting fqv's.

ii) Order the concepts in each elongated fqv in decreasing dv order.

iii) Retain top n concepts of the ordered vector, where n depends on the particular vector and the strategy used, as explained later.

iv) Reorder the shortened vector back to the original ordering of the concepts.

v) Do the original iteration of document retrieval search using the trimmed fqv's and compare its performance to that of the regular feedback from step (i) above.

The motivation for this approach is as follows. In Murray's work, the lowest weight concepts are shown to have small correlation ratios and discarding such terms is not considered to be harmful. On the other hand, the theory of dv's of terms and specifically their non-monotonic relationship with frequency of occurrence of the corresponding terms, as shown by Crawford [4], indicates that some presently low weight but good discriminating terms could be of potential importance in the present and future correlating process. Thus, throwing away such potentially valuable terms might hurt the retrieval performance, while retaining them might really help.

4. Vector Trimming Strategies

First, some notations are given and operations are defined:

i) Notations:

a) $X \xrightarrow{0} Y$: Operator 0 operates on the initial vector X to yield the final vector Y.

b) V : Elongated fqv which is to be reduced in length.

c) A-order : Original alphabetical order (of the concept numbering). Thus a vector in A-order means that its concepts are numbered in original alphabetical order.

d) D-order : Decreasing dv order (of the concept numbering). Thus a vector in D-order means that its concepts are numbered in decreasing dv order.

ii) Possible Operators, 0:

a) A/D : Concepts of the initial vector are renumbered from A-order to D-order to yield the final vector.

b) D/A : Concepts of the initial vector are renumbered from D-order to A-order to yield the final vector.

c) Murray n : From the initial vector, top n% of the total number of concepts in their decreasing weight order are retained to form the final vector.

d) Fixed n : The final vector is composed of fixed top n concepts of the initial vector. If the length of the initial vector is less than n, then both the final and the initial vectors are identical.

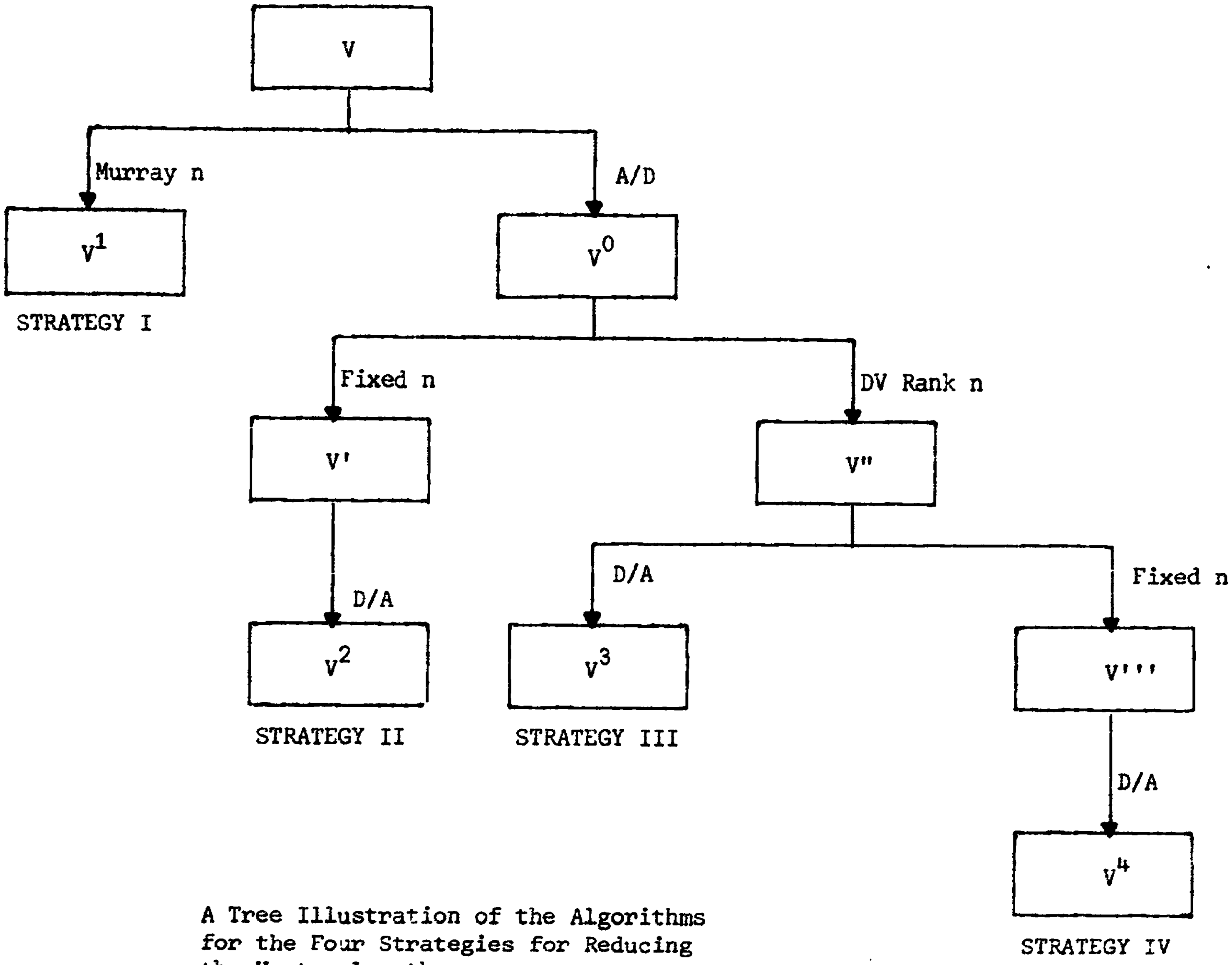
e) DV Rank n : The final vector is composed of all concepts of the initial vector with dv rank (to be defined shortly) less than or equal to n.

Utilizing these notations and operators, Fig. 1 is a tree illustration of the algorithms of all the four strategies for reducing the vector lengths. Assuming the original query vectors, Q of Fig. 2(a) for illustration purposes, a brief description of each of the strategies follows:

A) Strategy I

As described earlier, this is Murray's [2] strategy and for every vector V, it retains the top n% of the total number of concepts in their decreasing weight order to produce a shortened length vector V^1. Suggested value of n is around 20. Fig. 3 shows the vectors V^1 obtained from vectors V of Fig. 2(b), by 70% reduction in length.

For the following three strategies, the first common step is to renumber the concepts of each elongated vector V from their A-ordering to D-ordering.



A Tree Illustration of the Algorithms
for the Four Strategies for Reducing
the Vector Lengths

Fig. 1

```
Query #  *FIND   Query #                 # of Concepts       No. of relevant documents
Concept weight pairs
Relevant Document Numbers
```

(i) Format of Each Vector

```
   4    *FIND  4
1161   12 2779  12 6869  12 7248  1211433  12                                          5    8
 123   124  126  127  131  132  133  135
   5    *FIND  5
 188   24  878  12 2740  12 4152  24 4194  12 6714  12 8467  12 8469  12               8   12
  31    32   33   34   52   53   54   56   57   58   59   60
```

(ii) Original Query Vectors, Q, with concepts in A-order.

(MEDLARS Collection)

Fig. 2(a)

```
   4    *FIND  4                                                                   61     8
 235   72  502   12  556   12  670   12  682   12 1000   12 1161   12 1165   12
2172   84 2235   12 2261   12 2779  144 3065   24 3161   12 3916   36 3919   12
4026   12 4210   24 5161   72 5378   12 5634   12 5715   12 5788   24 5831   24
5840   12 5972   12 6096   24 6098   12 6484   12 6761   12 6935   24 6950   12
6971   12 6972   48 7089   12 7214   24 7678   12 7725   24 8031   12 8222   36
8767   12 8830   12 8881   24 9370   12 9391   12 9400   12 9493   12 9681   12
10952  1210604   1210660   2410661   1 10741   1211433   2411511   1211529   12
11628 12011973   1211981   1212004   2412005   12
 123   124   126   127   131   132   133   135
   5    *FIND  5                                                                  264    12
 155   12  188 420  208   24  217   12  250   12  270   12  385   12  432   12
 447   12  596   12  603   24  656   12  690   12  758   12  826   24  829   24
 857   12  878   12  908   12  996   12 1013   48 1036   12 1042   84 1128   12
2071   12 2072   24 2085   48 2105   12 2172   12 2191   12 2226   12 2231   12
2253   12 2298   60 2321   24 2345   12 2425   12 2440   12 2486   12 2502   24
2512   12 2522   24 2524   12 2541   12 2587   12 2594   48 2607   12 2610   12
```

Feedback Query Vectors, V, with concepts in A-order.

(MEDLARS Collection)

Fig. 2(b)

```
    4    *FIND  4                                                                   60    8
    6   12    10   12    17   12    24   72    35   12    70   12    98   24   128   24
  170   72   171   12   173  144   232   48   270   12   307   24   308   12   330   36
  404   12   453   24   559   24   577   12   592   12   639   12   723   12   726   36
  730   12  1009   12  1113   12  1161   12  1174   12  1247   12  1278   12  1300   12
 1526   12  1573   12  1617   12  1746   24  2216   12  2221   12  2343   24  2484   12
 2803   12  3993   12  4118   12  4124   12  4128   12  4137   1212869    1212885   12
12891   1212896    1212922    2412941    2412952    2412954    1212960    1212971   12
12978 12012980    1212983    2412997   24
  123   124   126   127   131   132   133   135
    5    *FIND  5                                                                  262   12
   16  180    28  288    45   12    56   12    62   12    80  108    82  168    90   12
  101   12   104   12   121   96   139  108   168   48   177  132   188   84   195   24
  196   48   213   12   241   12   247   12   252   12   253   36   269   12   310   24
  318   48   326   12   358   60   395   60   398   12   403   96   410   84   417   24
  428   48   429   12   442   48   461   12   468   24   490   72   540   12   550   60
  592   12   595   12   600   36   609   24   610   84   628   12   639   96   659   96
  666   12   672   12   676   48   689   12   698   12   702   24   715   48   726   24
  727   72   775   24   776   12   784   12   809   36   813   12   816   36   830   36
```

Feedback Query Vectors, V^o, with concepts in D-order.

(MEDLARS Collection)

Fig. 2(c)

```
    4    *FIND  4                                                                      18    8
  235   72 2172   84 2779  144 3916   36 5161   72 5788   24 5831   24 6096   24
 6935   24 6972   48 7214   24 7725   24 8222   36 8881   2410660    2411433   24
11628  12012004   24
  123   124   126   127   131   132   133   135
    5    *FIND  5                                                                      79   12
  1 3  420 1013   48 1042   84 2085   48 2298   60 2594   48 2637   72 2930   36
 2975   96 3866   48 4152  336 4193   36 4194   96 4375  168 4591  288 5344   48
 5746  204 5772   36 5813  180 5826   36 5873   24 5878  132 5978   24 5996   48
 6577   60 6580   48 6586   48 6593   48 6714  120 6737   48 6770   84 6777   36
 6852   24 7047  108 7170   48 7171   48 7196   24 7326   72 7403   60 7661   48
 7690   24 7726   48 7791   24 7836   36 8022   24 8111   24 8218   72 8222   24
 8402   36 8425   36 8460   24 8467   48 8468   24 8469   36 9478   72 8570   36
 8692   96 8695   36 8746   24 8922   36 9370   24 9379   24 9500   48 9590   24
 9681   9610230    3610422    2410520    2410680    8410802    2411433    2411472   48
11571   2411666    2411684    3611726    2411859    9612169    3612170  108
   31    32    33    34    52    53    54    56    57    58    59    60
```

Reduced Length Feedback Query Vectors, V^1, with concepts in A-order -- Strategy I.

(MEDLARS Collection 30% concepts retained)

Fig. 3

Call the modified vector V^o. Fig. 2(c) shows the vector V^o obtained from vectors V.

B) Strategy II

This strategy places a fixed upper bound on the length of each of the reduced vectors. This yields standardization of the lengths of vectors and could make programming a little easier and more efficient.

A fixed number n is chosen; top n best discriminating concepts (if they exist, otherwise all) of V^o (which is in D-order) are retained and renumbering of the concepts to A-ordering is done to achieve the reduced length vector V^2. Figs. 4(a) and 4(b) depict the last two steps in obtaining vector V^2 from V^o, for $n = 30$.

Suggested value of n is the average length of the document vectors in the document collection.

C) Strategy III

Here, the idea is to retain all those concepts which are the best discriminators. Specifically, dv's of all concepts present in the document collection are calculated using Crawford's methods [4] and the concepts are ranked in D-order. The rank of a concept in such an ordering is called *dv rank* of the concept. Thus, for example, any concept number in vectors of V^o is equal to its dv rank.

A dv-dv rank curve is plotted for the collection. The curve is an exponential looking curve for most of its range (Fig. 5). This curve has a sharp drop in the beginning and then approaches the X-axis asymptotically before it goes negative very steeply.

```
   4    *FIND   4                                                                    30    8
   6   12    10   12    17   12    24   72    35   12    70   12    98   24   128   24
 170   72   171   12   173  144   232   48   270   12   307   24   308   12   330   36
 404   12   453   24   559   24   577   12   592   12   639   12   723   12   726   36
 730   12  1009   12  1113   12  1161   12  1174   12  1247   12
 123   124   126   127   131   132   133   135
   5    *FIND   5                                                                    30   12
  16  180    28  288    45   12    56   12    62   12    80  108    82  168    90   12
 101   12   104   12   121   96   139  108   168   48   177  132   188   84   195   24
 196   48   213   12   241   12   247   12   252   12   253   36   269   12   310   24
 318   48   326   12   358   60   395   60   398   12   403   96
  31    32    33    34    52    53    54    56    57    58    59    60
```

Reduced Length Feedback Query Vectors, V', with concepts

in D-order -- Strategy II.

(MEDLARS Collection - Fixed 30)

Fig. 4(a)

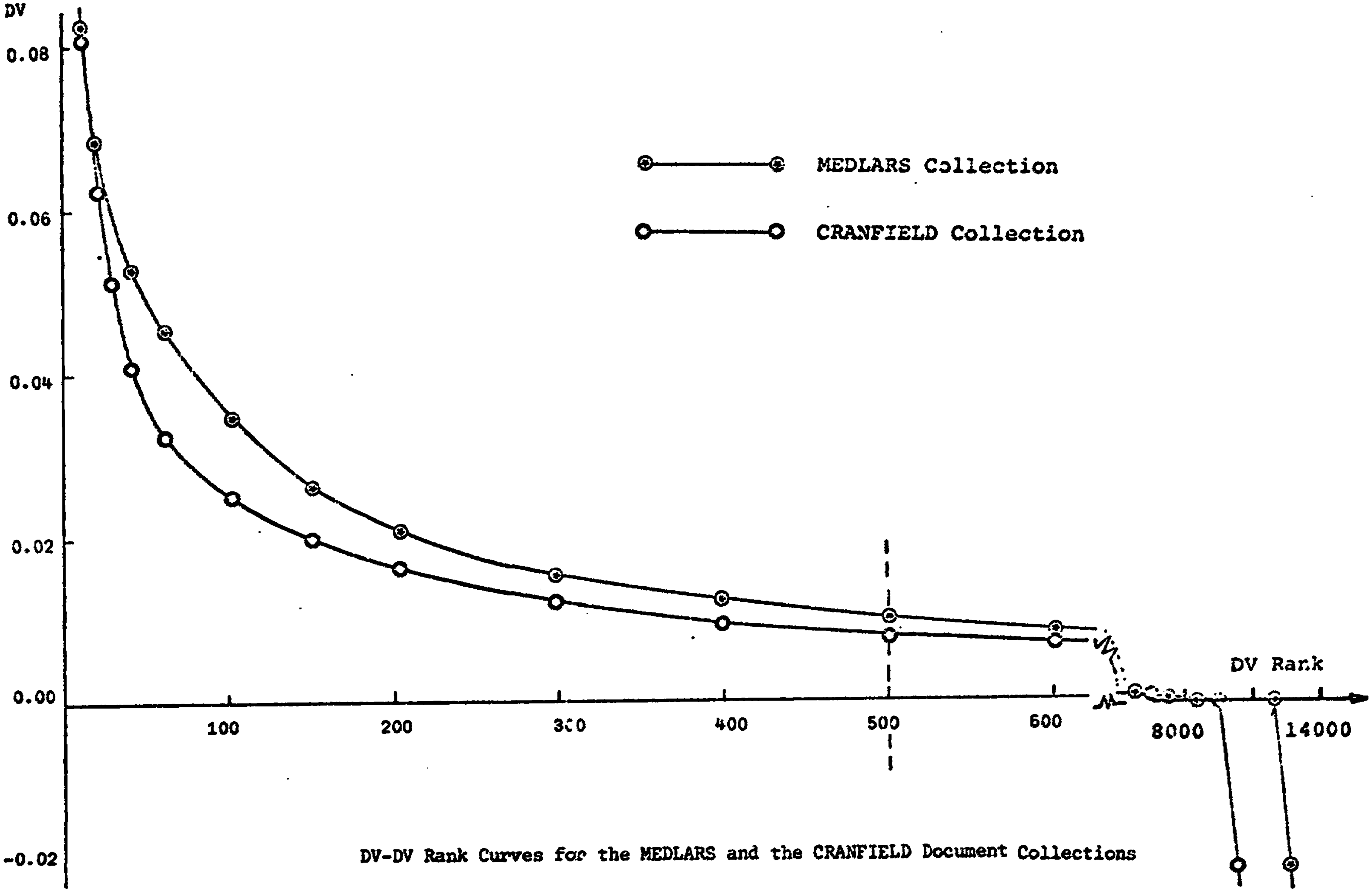
```
   4    *FIND   4
                                                                                    30     8
 235    12  556   12   670   12 1000   12 1161   12 2261   12 2779  144 3916   96
3919    12 5161   12  6096   24 6098   12 6761   12 6935   24 6950   12 6972   48
7214    24 7678   12  8031   12 8222   36 9681   1210052   1210604   1210660   24
11511   1211529   1211973   1211981   1212004   2412005   12
 123   124   126   127   131   132   133   135
   5    *FIND   5
                                                                                    30    12
 108    12 1036   12  2524   12 2587   12 2975   96 3081   12 3320   12 3872   12
4197    12 4375  168  4591  288 4666   12 5782   24 5813  180 5878  132 6577   60
6590    48 6598   48  6770   84 7047  108 7132   12 7403   60 7661   48 8692   96
8695    36 9598   1210220   1210300   1210520   2412170  108
  31    32    33    34    52    53    54    56    57    58    59    60
```

Reduced Length Feedback Query Vectors, V^2, with concepts

in A-order -- Strategy II.

(MEDLARS Collection - Fixed 30)

Fig. 4(b)



DV-DV Rank Curves for the MEDLARS and the CRANFIELD Document Collections

Fig. 5

A value m for dv rank cutoff is determined from the curve; all concepts in V^o with dv rank $\leq m$ are retained and then final renumbering of the concepts from D-order to A-order yields the final reduced length vector V^3. These last two steps for obtaining V^3 from V^o are exhibited by vectors of Figs. 6(a) and 6(b).

A recommended value of m, the dv rank cutoff, is the one near the foot of the first steep of the dv-dv rank curve. More exactly, that value of m is chosen where the slope of the curve is $\leq \varepsilon$, where ε is a constant for the particular collection. A value of $\varepsilon = 0.00004$ giving $m = 500$ is found appropriate for the document collections used in this project. A method for determining ε for a particular document collection is given later in Section 6(B).

D) Strategy IV

This strategy is an intermediate between the last two strategies and using it the length of any reduced length vector is the minimum of the lengths obtained by using strategies II and III. It is same as the previous strategy except that the length of each vector is further trimmed to n just before the concepts are renumbered back to A-ordering. Thus, in addition to using a dv rank cutoff m like Strategy III, it also places an upper bound n on the length of each vector like Strategy II, to yield the final reduced length vector V^4 (Figs. 7(a) and 7(b)).

Note that this strategy is also equivalent to using Strategy II with a maximum fixed upper bound on lengths being equal to n, and in addition, trimming the vectors further down such that all concepts with dv rank greater than m are eliminated.

```
  4    FIND  4                                                        18    8
  6   12   10   12   17   12   24   72   35   12   70   12 .  98   24  178   24
170   72  171   12  173  144  232   48  270   12  307   24  308   12  330   36
404   12  453   24
123  124  126  127  131  132  133  135
  5    FIND  5                                                        38   12
 16  120   23  288   45   12   56   12   62   12   80  108   82  168   90   12
101   12  104   12  121   96  139  108  168   48  177  132  188   84  195   24
196   48  213   12  241   12  247   12  252   12  253   36  269   12  310   24
313   48  326   12  358   60  395   60  398   12  403   96  410   84  417   24
 28   48  429   12  442   48  461   12  468   24  490   72
 31   32   33   34   52   53   54   56   57   58   59   60
```

Reduced Length Feedback Query Vectors, V", with concepts

in D-order -- Strategy III

(MEDLARS Collection - DV Rank 100)

Fig. 6(a)

```
  4    FIND  4                                                        18    8
 235   72  576   12 2779  144 3516   36 5019   12 5161   72 6761   12 6935   24
6972   48 7214   24 7578  1210052  1210434  1210660  2411511  1211973   12
11541  1212004   24
123  124  126  127  131  132  133  135
  5    FIND  5                                                        38   12
 563   12 1013   48 1036   12 2524   12 2557   12 2637   72 2975   96 3081   12
3370   12 3372   12 4197   12 4375  168 4591  288 4666   12 5216   24 5782   24
5813  120 5878  132 6577   60 6580   48 6638   48 6770   84 7047  108 7132   12
7171   48 7403   60 7761   48 8760   24 9592   96 9595   36 9938   12 9998   12
10220  1210300  1210520  2410680  2411192  1212170 108
 31   32   33   34   52   53   54   56   57   58   59   60
```

Reduced Length Feedback Query Vectors, V^3, with concepts

in A-order -- Strategy III

(MEDLARS Collection - DV Rank 100)

```
  4    *FIND  4
                                                                            18    8
  6  12    10  12    17  12    24  72    35   12    70  12    98  24   128   24
170  72   171  12   173 144   232  48   270   12   307  24   308  12   330   36
404  12   453  24
123  124  126  127  131  132  133  135
  5    *FIND  5
                                                                            30   12
 16 180    28 288    45  12    56  12    62   12    80 108    82 168    90   12
101  12   104  12   121  96   139 108   168   48   177 132   188  84   195   24
196  48   213  12   241  12   247  12   252   12   253  36   269  12   310   24
318  48   326  12   358  60   395  60   398   12   403  96
 31   32   33   34   52   53   54   56   57   58   59   60
```

Reduced Length Feedback Query Vectors, V''', with concepts

in D-order -- Strategy IV

(MEDLARS Collection - DV Rank 500/Fixed 30)

Fig. 7(a)

```
  4    *FIND  4
                                                                            18    8
 235   72  556   12 2779 144 3916  36 3919  12 5161  72 6761  12 6935   24
6972   48 7214   24 7678  1210052  1210604  1210660  2411511  1211973   12
11981  1212004   24

 123  124  126  127  131  132  133  135
   5   *FIND  5
                                                                            30   12
 308   12 1036   12 2524  12 2587  12 2975  96 3081  12 3320  12 3872   12
4197   12 4375  168 4591 288 4666  12 5782  24 5813 180 5873 132 6577   60
6530   48 6598   48 6770  84 7047 108 7132  12 7403  60 7661  48 8692   36
8695   36 9598  1210220  1210300  1210520  2412170 108
  31   32   33   34   52   53   54   56   57   58   59   60
```

Reduced Length Feedback Query Vectors, V'', with concepts

in A-order -- Strategy IV

(MEDLARS Collection - DV Rank 500/Fixed 30)

Fig. 7(b)

As compared to the last strategy, the advantage of using this strategy is seen in getting standardized fixed length vectors, although some loss in performance is expected due to the loss of good discriminators (upto dv rank cutoff m) in some of the vectors.

5. Experimental Environment

A) Retrieval System

The SMART automatic document retreival system is used as the test bed for conducting the experiments. This retrieval system is a major facility for conducting experiments to test and evaluate various document retrieval strategies.

B) Data Collections

Two different SMART document collections used in the present series of experiments are MEDLARS and CRANFIELD. The MEDLARS collection used is a 450 document subset of the originally larger collection dealing with varied medical literature; the associated number of queries is 30. The indexing procedure used for representation of the documents and the queries in the machine utilizes a word stem dictionary.

On the other hand, the CRANFIELD collection consists of a more homogeneous set of 424 documents dealing with aerodynamics; the associated queries form a 125 query subset of originally 155 query collection. The indexing process used for this collection makes use of a word form dictionary. The procedure of indexing making use of dictionaries has been discussed in detail by Salton [7].

In many respects, these two document collections are different enough to warrant putting a great deal of confidence into the results based on the experiments done on them.

C) Clustering Parameters

The experiments are conducted on clustered document collections because Murray [2] has shown that document retrieval based on clustered files is more efficient than the one based on inverted files or individual documents, for instance.

A clustered CRANFIELD document collection is available as a SMART collection, while the MEDLARS collection was clustered for this experiment. The clustering parameters used are:

i) SMART routine CLUSTER is used for clustering by Rocchio's algorithm [8].

ii) The loose documents are placed with the centroid with which they correlate the highest.

iii) The algorithm is allowed to choose an optimum number of documents to be batched for checking as possible cluster roots.

iv) For Rocchio's density test, to be a cluster point, at least 4 documents must have a correlation greater than 0.3 and at least 8 documents must have a correlation greater than 0.1 with it.

v) Minimum and maximum number of documents in each cluster are 8 and 25 respectively, excluding the loose documents blended in later on, as determined by step (ii) above.

D) Searching Parameters

The SMART routine SEARCH is used for document searching experiments. The parameters used throughout this study are as follows:

i) Feedback Parameters

a) The number of documents retrieved for each iteration is 30.

b) Among the top 10 relevant documents, all the relevant ones are added to and the top nonrelevant subtracted from the original query to form the first iteration query. Only one iteration of feedback is carried out for the present experiments.

ii) Cluster Searching Parameters

a) At least 40 documents are correlated with the query on each iteration.

b) At least 3 and at most 10 cluster nodes are expanded for each iteration.

c) Any nodes whose cosine correlation with the query is within 0.01 of the latest node selected for expansion are also expanded.

E) Evaluation Techniques

The basic evaluation technique used for the comparison of various retrieval search runs, is the Precision-Recall (P-R) curve [7]. Even though none of the fluid or frozen feedback searches provide the true retrieval performance while the more exact test and control group feedback method [9] is time consuming, the fluid searches are chosen with the intention of making relative comparisons only.

The "ranking" and "feedback" effects occurring in relevance feedback, as discussed by Hall and Weiderman [10] are analyzed manually. Particularly, the "feedback effect", which measures the improvement in retrieval performance due to the new relevant retrieved documents, is considered.

6. Experimental Details

A) Overall Flowchart of the Experiments

Fig. 8 is an overall flowchart of the document retrieval search experiments. Basically, as discussed in Section 3(C) also, an experiment to test out a strategy consists of the following four steps:

i) Perform document retreival SEARCH (SMART routine) on a collection, using one Rocchio-type feedback iteration. Obtain the P-R curves for the original (ORIG) and feedback (FDBK) iterations.

ii) Shorten the elongated fqv's using one of the Vector Trimming Strategies to get the reduced length vectors.

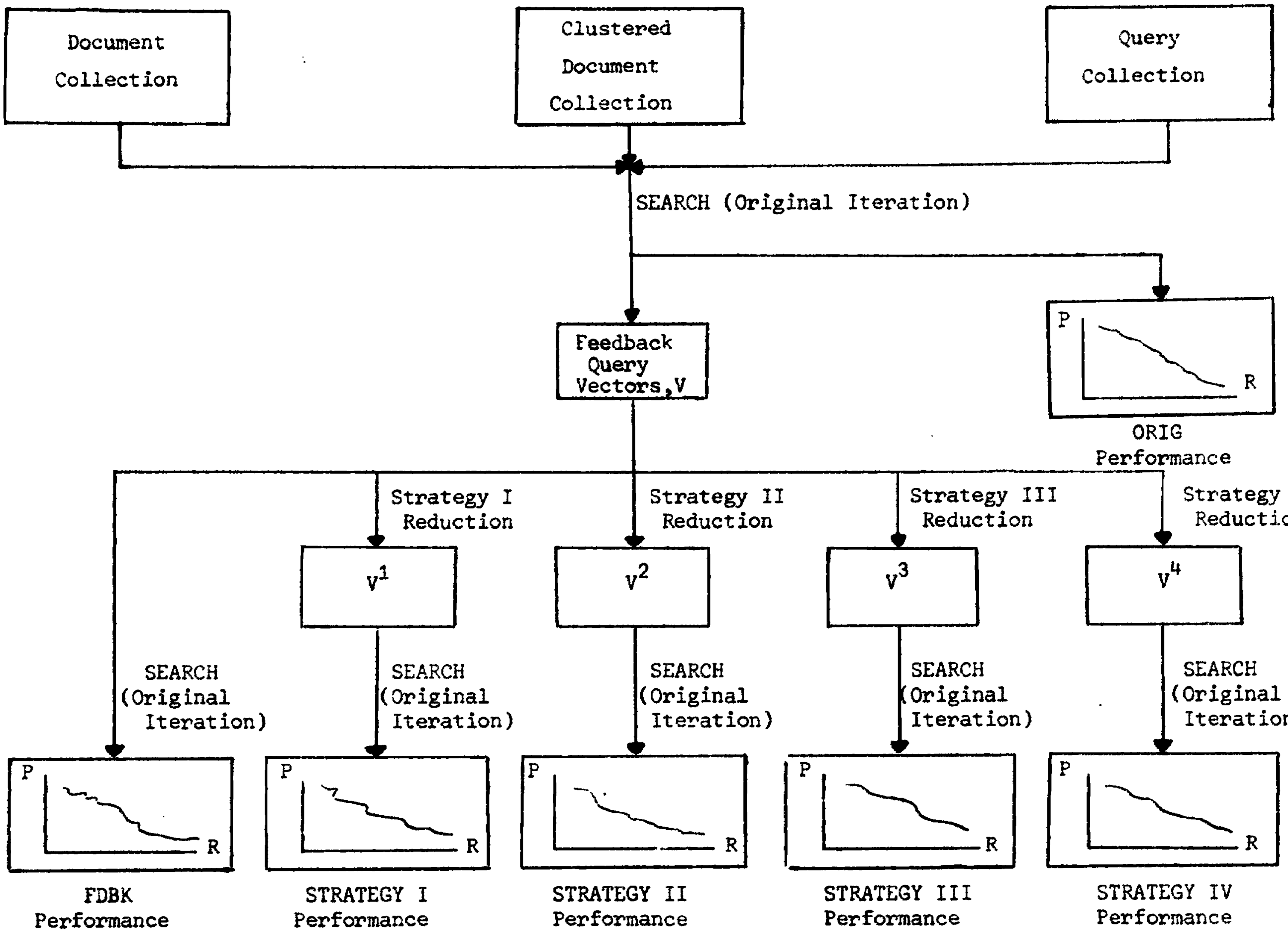
iii) Use these modified fqv's to SEARCH the document collection without any further feedback. Obtain the P-R curve for this modified (MOD) iteration.

iv) Compare the SEARCH results for the FDBK and the MOD iterations.

B) Detailed Description of One of the Experiments

To give an idea of how these experiments are performed down to their inner det..ls, a detailed description of one of the experiments is presented.

Table 1 details the steps of the experiment using the MEDLARS document collection and Strategy III for the length reduction of the vectors. Figs. 2 and 6 give computer output examples of the vectors at various stages through the experiment.



Overall Flowchart of the Document Retrieval Search Experiments Performed to test Various Strategies for Reducing the Vector Lengths.

Fig. 8

STEP NO.	INPUT	ROUTINES USED and/or OPERATIONS PERFORMED	OUTPUT
1	a) MEDLARS Document collection, C b) MEDLARS Query collection, Q (Fig. 2A)	SMART routine SEARCH: original (ORIG) and the first feedback (FDBK) iterations are performed	i) P-R curves O and F for ORIG and FDBK iterations. ii) Punched feedback query vectors, V (Fig. 2B)
2	C	Crawford's Discrimination Program [4]: gives the list of concepts in D-order. Output format is made suitable for use in step #7 below.	List L
3	L ϵ(a predetermined parameter, constant for a particular collection).	A curve showing dv vs. dv rank of concepts is plotted. The value of dv rank cutoff, m, where slope of the curve is $\leq \epsilon$, is determined.	DV rank cutoff, m (Fig. 5)
4	L	FORTRAN program ALPHDV: gives the mapping of concepts from A-order to D-order. Output format is made suitable for the next step.	Mapping M.
5	V M	SMART routine RECODE: changes concept numbers in the definition of fqv's from A-order to D-order	Renumbered fqv's, v^o, obtained as $v \xrightarrow{A/D} v^o$ (Fig. 2C)
6	V^o m	FORTRAN program RETAIN: For each vector of v^o, it retains only those concepts with their concept number (= dv rank) $\leq$ m.	Reduced length fqv's, V'' , in D-order (Fig. 6A)
7	V'' L	SMART routine RECODE: restores A-order of concepts within each vector of V'' .	Reduced length fqv's, v^3, in A-order (Fig. 6B)
8	V^3 C	SMART routine SEARCH: original iteration with no feedback is performed for document searching	P-R curve, F^3, showing performance of reduced length fqv's
9	F F^3	Compare P-R curves and other retrieval statistics.	Get the results.

Detailed Description of the Experiment to implement Strategy III on the SMART system.

Table 1

For this part of the experiment, the value of parameter ε was first determined for the MEDLARS collection by trial and error, such that the dv rank cutoff lies at the foot of the first sharp drop of the dv-dv rank curve (Fig. 5). The value of ε is found to be 0.00004 and the same value of ε is used for the CRANFIELD collection. This value is determined just once for the whole collection.

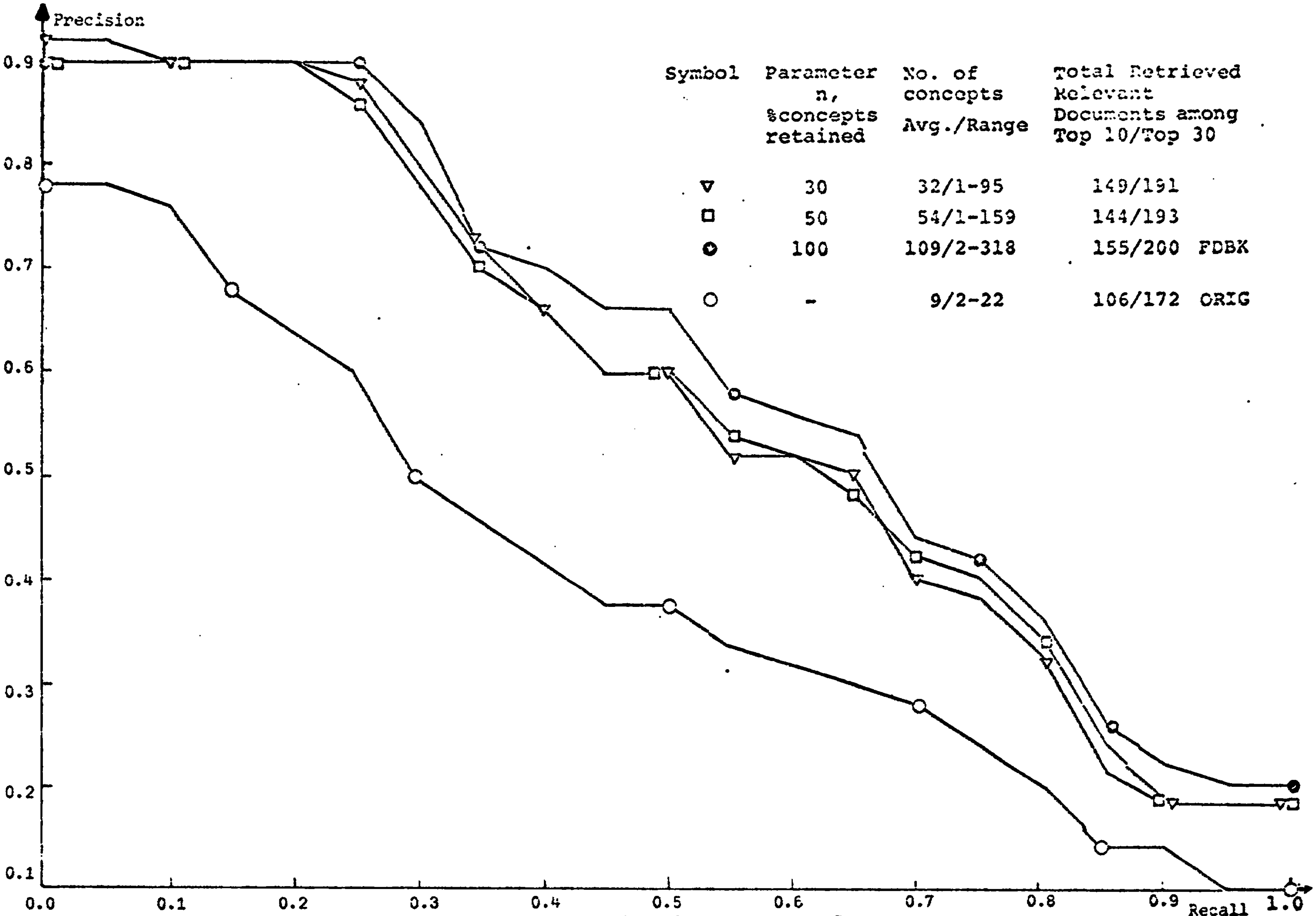
Another important point about this experiment is that during the length reduction of the vectors, care was taken to prevent the query from getting zeroed out completely. Thus, if after the application of the strategy, the reduced query vector happened to contain no concept at all, the algorithm took care that at least 5 concepts (if present in the original query vector, otherwise equal to number of concepts in it) were retained. Similar precaution was taken during the application of other strategies as well.

7. Results

A) Performance Curves Obtained

First, the effect of variation of individual parameters n and m (n and m are the parameters used in the description of various strategies in Section 4) is studied for each strategy separately and performance curves are obtained. The best curve obtained for each strategy is taken and comparisons are made between these.

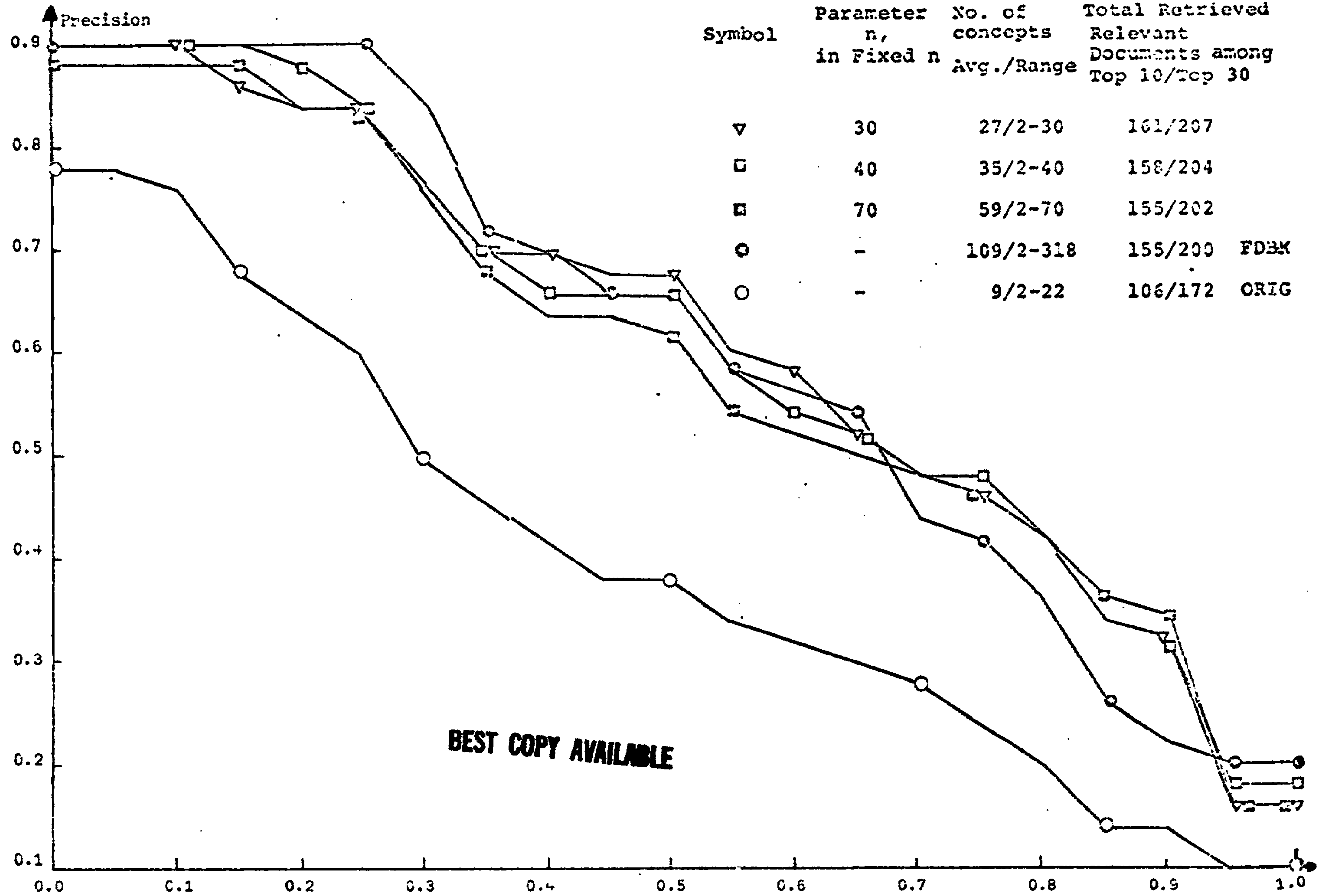
Figs. 9(a) to 9(d) show the performance curves obtained for each individual of the four strategies, using the MEDLARS document collection. Fig. 9(e) shows the comparison of the best P-R curves -- one obtained for each of the strategies. Furthermore, in each of these figures, the average and the range of number of concepts present in the fqv's and the total number



Symbol	Parameter n, %concepts retained	No. of concepts Avg./Range	Total Retrieved Relevant Documents among Top 10/Top 30	
▽	30	32/1-95	149/191	
□	50	54/1-159	144/193	
●	100	109/2-318	155/200	FDBK
○	-	9/2-22	106/172	ORIG

Performance Curves for the MEDLARS Collection -- Strategy I

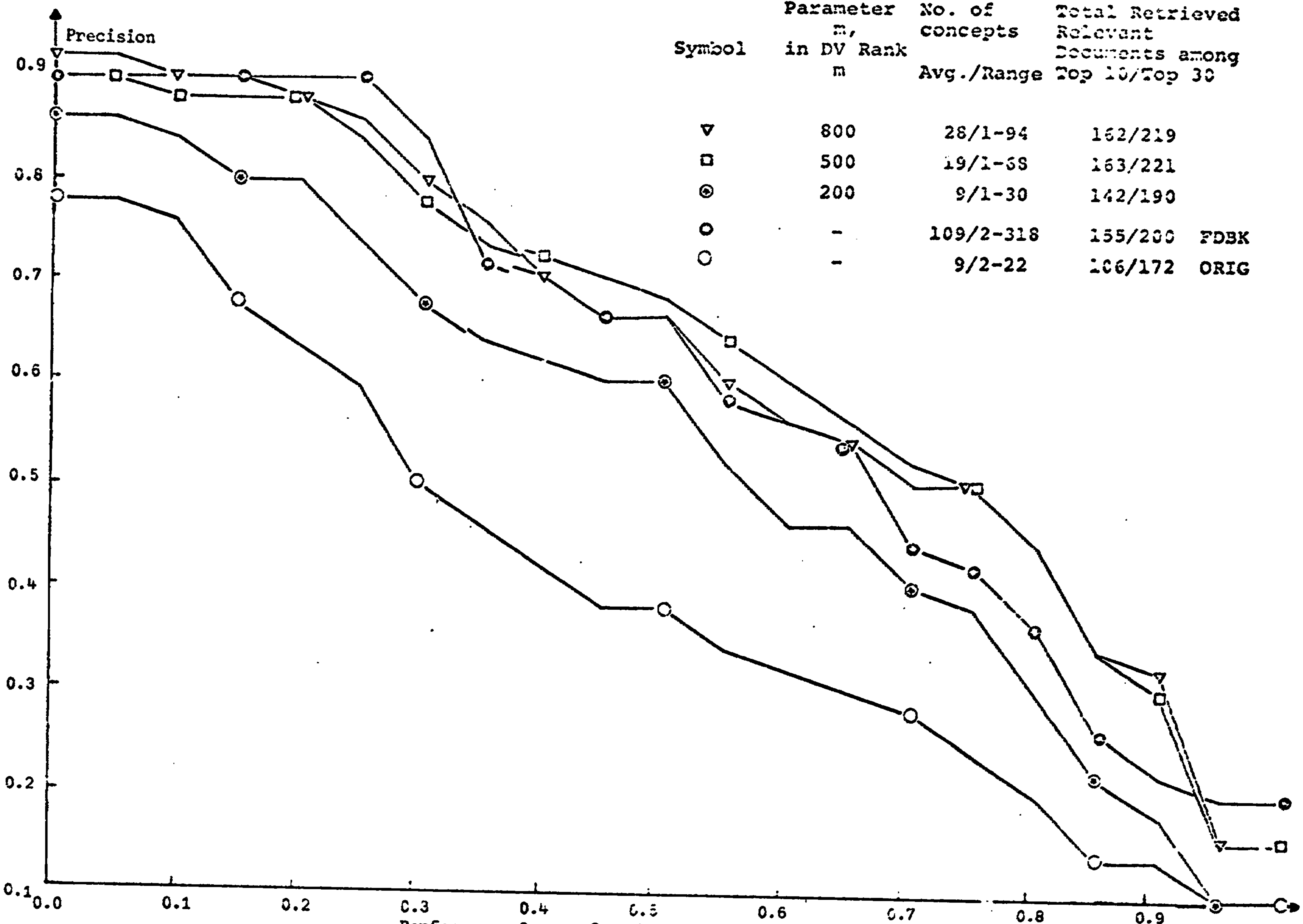
Fig. 9(a)



Symbol	Parameter n, in Fixed n	No. of concepts Avg./Range	Total Retrieved Relevant Documents among Top 10/Top 30	
▽	30	27/2-30	161/207	
□	40	35/2-40	158/204	
⊡	70	59/2-70	155/202	
●	-	109/2-318	155/200	FDBK
○	-	9/2-22	106/172	ORIG

Performance Curves for the MEDLARS Collection -- Strategy II.

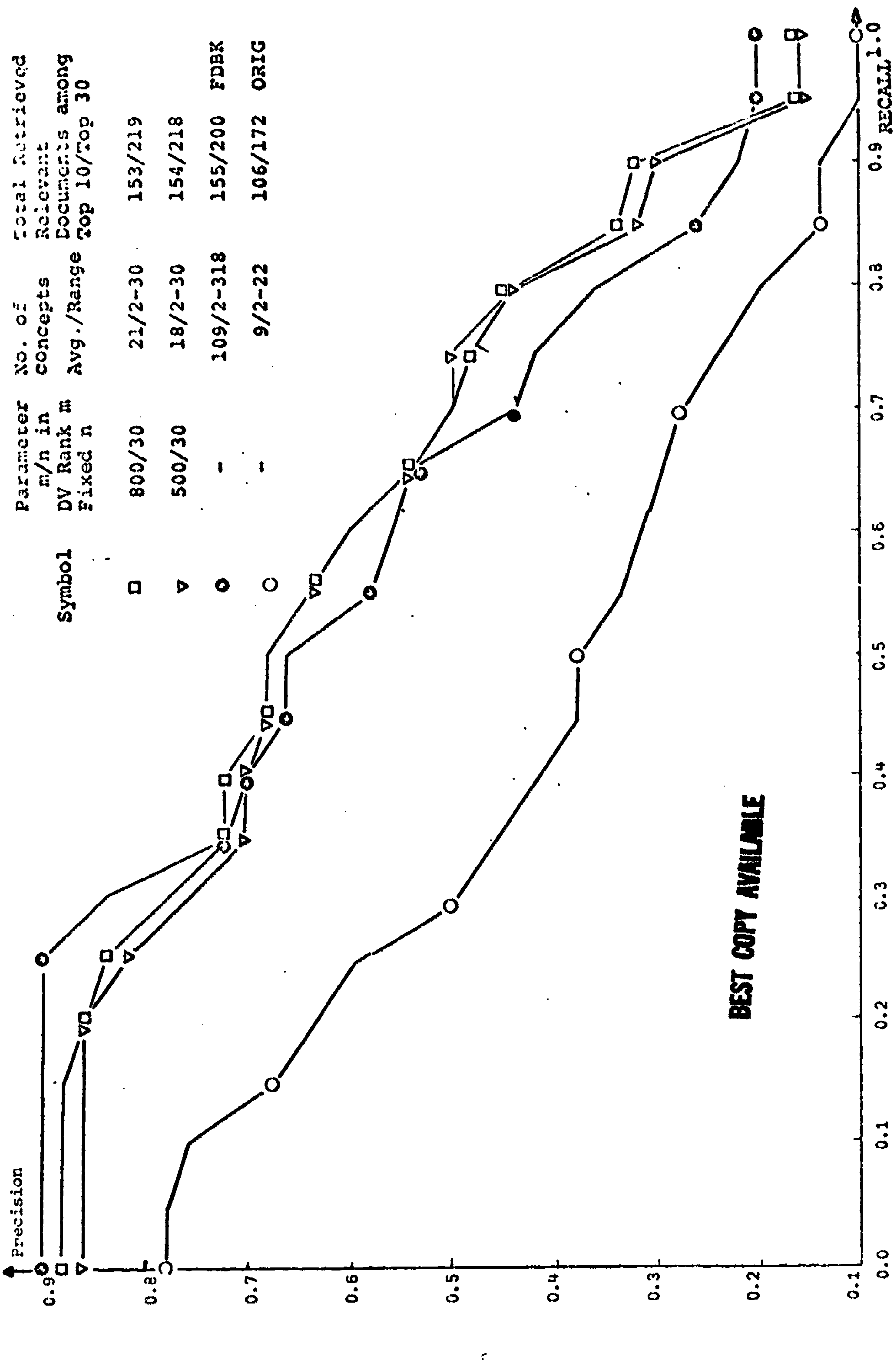
Fig. 9(b)



Symbol	Parameter m, in DV Rank m	No. of concepts Avg./Range	Total Retrieved Relevant Documents among Top 10/Top 30	
▽	800	28/1-94	162/219	
□	500	19/1-58	163/221	
⊙	200	9/1-30	142/190	
◯	-	109/2-318	155/200	FDBK
○	-	9/2-22	106/172	ORIG

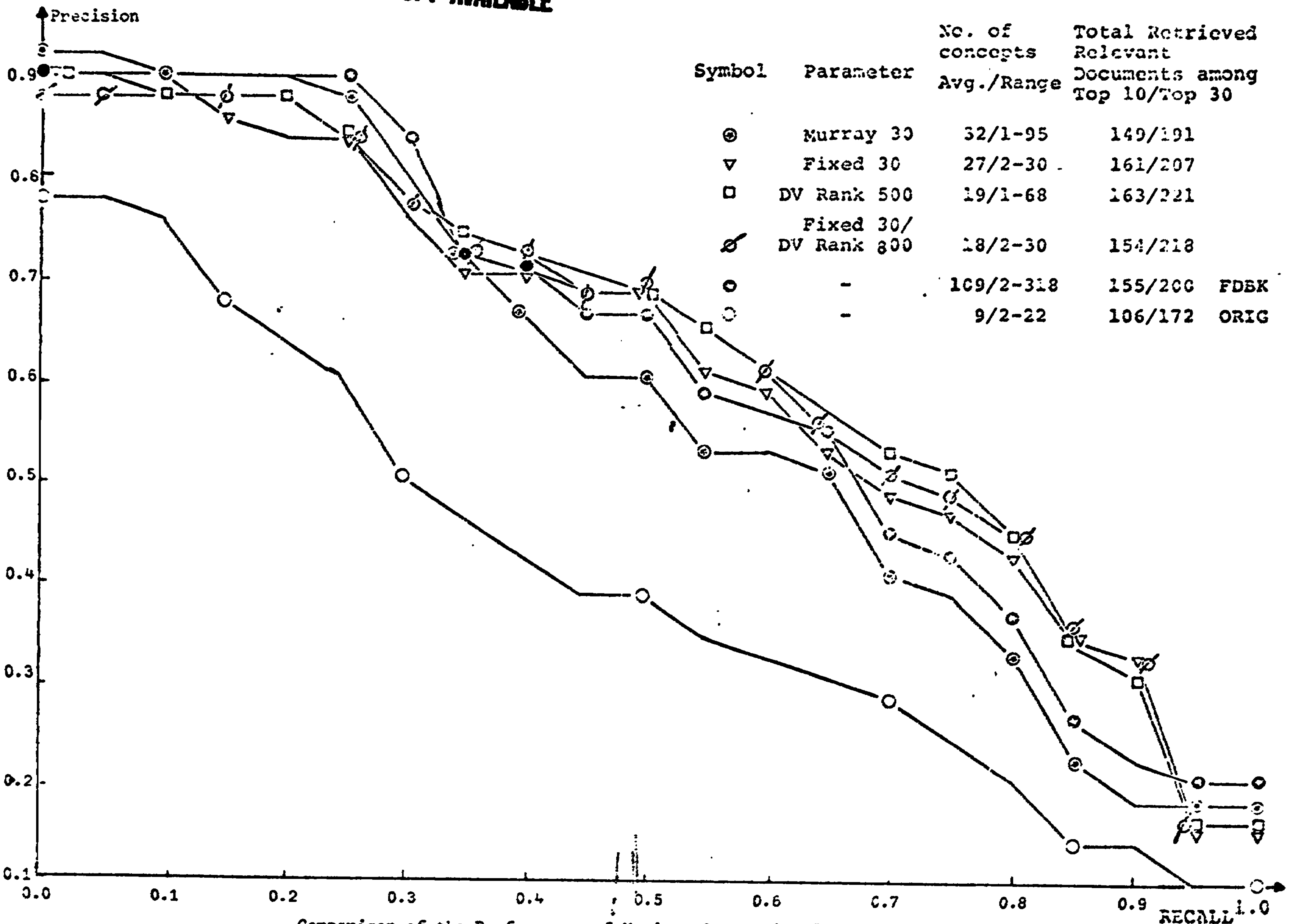
Performance Curves for the MEDLARS Collection -- Strategy III

Fig. 9(c)



Performance Curves for the MEDLARS Collection -- Strategy IV

Fig. 9(d)



Symbol	Parameter	No. of concepts Avg./Range	Total Retrieved Relevant Documents among Top 10/Top 30	
⊛	Murray 30	32/1-95	149/191	
▽	Fixed 30	27/2-30	161/207	
□	DV Rank 500	19/1-68	163/221	
ø	Fixed 30/ DV Rank 800	18/2-30	154/218	
○	-	109/2-318	155/200	FDBK
◌	-	9/2-22	106/172	ORIG

Comparison of the Performance of Various Strategies for the MEDLARS Collection

Fig. 9(e)

of relevant documents retrieved among the top 10 and the top 30 ranks for the whole query collection are also given. For comparison purposes, the P-R curves obtained for ORIG and FDBK iterations are included in each of the figures.

Fig. 10 shows the comparison of the best P-R curves obtained for the various strategies for the CRANFIELD collection, except that Strategy IV was not tried, because results of using this strategy were expected to be similar to those obtained for the MEDLARS collection.

B) Inference from the Performance Curves

In the following analysis, all comparisons are made with respect to the performance curve obtained by the first regular feedback (FDBK) iteration.

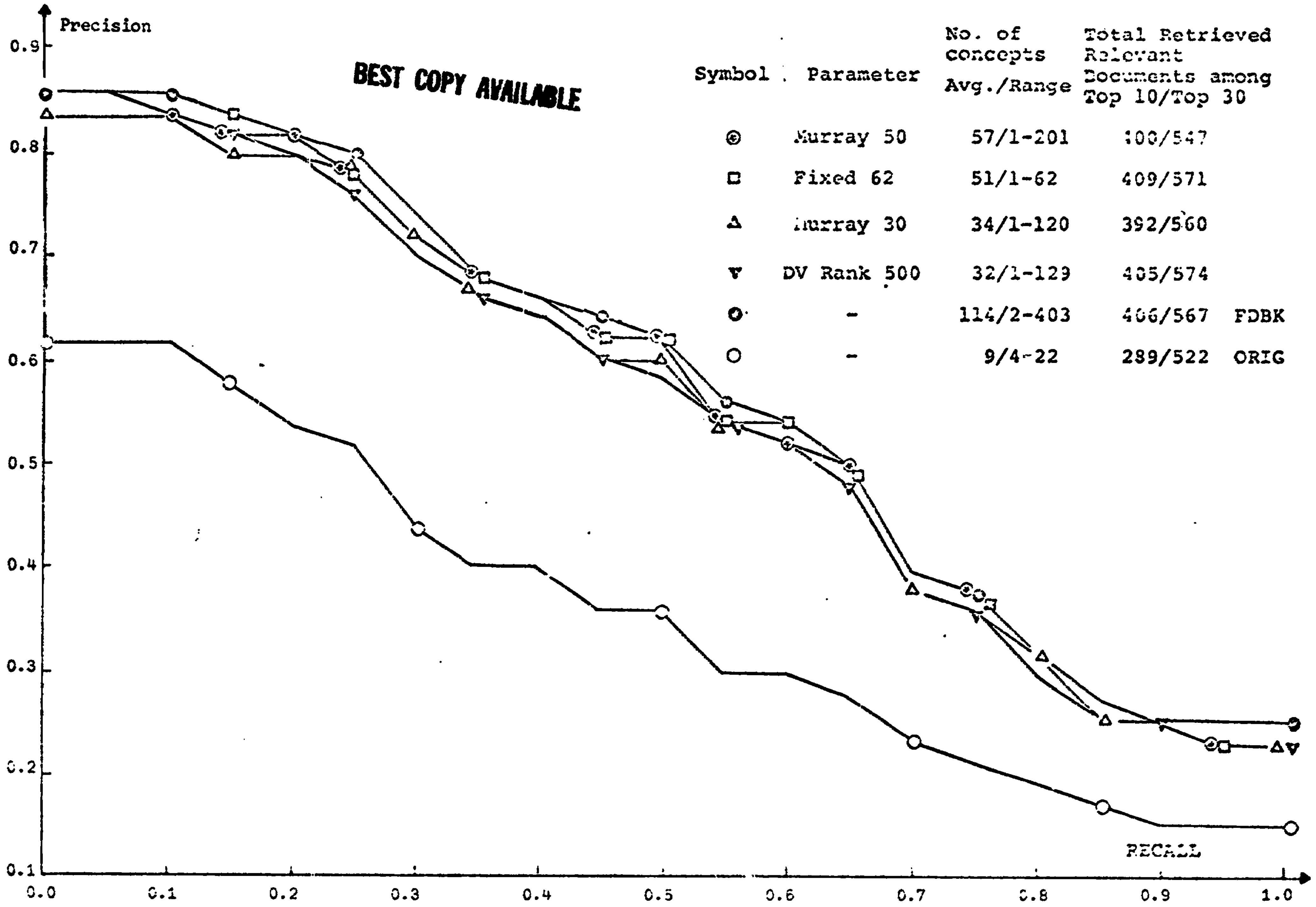
Tables 2, 3, 4 and 5 give such a comparative analysis for the four individual Strategies I, II, III, and IV, for the two collections used. Inference from these tables for each of the strategies is given below:

i) Strategy I

The use of this strategy for the reduction in the length of the fqv's results in almost consistent loss of performance by as much as 0.06 in precision at any recall level.

ii) Strategy II

Reduction in the lengths of the fqv's, using this strategy, results in almost equivalent or better performance than that obtained by using elongated fqv's. Keeping maximum fixed length of shortened fqv's equal to the average length of the document vectors seems reasonable and gives almost the best results, for high recall values, which is most likely what the user desires. Using this formula, the length



Comparison of the Performance of Various Strategies for the CRANFIELD Collection

Fig. 10

SERIAL NO.	COLLECTION NAME	
	MEDLARS (Fig. 9(a))	CRANFIELD
1	Retaining top 30% concepts for each fqv, that is n = 30, gives high precision for lower recall values	Retaining top 50% concepts for each fqv gives the best precision for all recall values, among all the reduced length vectors tried
2	For high recall values n = 50 gives the best results.	Vectors, with n = 20 give a P-R curve worse by 0.02 to 0.04 in precision for all recall levels.
3	Overall results are poorer than regular feedback by up to 0.06 in precision for same recall, for all the trimmed vectors tried.	Overall results are poorer than regular feedback by up to 0.04 in precision for any recall level, for all the vectors tried.

Performance Analysis of Strategy I in Comparison to the Regular Feedback Performance

Table 2

SERIAL NO.	COLLECTION NAME	
	MEDLARS (Fig. 9(b))	CRANFIELD
1	Initial value chosen for n is 40, being the recommended average length of the document vectors.	Correspondingly, initial value of n = 62 is taken, being the average length of the document vectors in the collection.
2	Values 30, 40 and 70 for n are tried. For all n the P-R curves are better than the regular feedback curve for higher recalls, which is probably what the user desires. The increase in precision at same recall is as much as 0.12.	n = 20 is the only other cut-off tried.
3	At low recalls, P-R curves for all values tried for n are worse than regular feedback curve by as much as 0.08 in precision.	For n = 62, the performance curve is almost equivalent to the regular feedback curve, while the average number of concepts in query vectors drops from 114 to 51.
4	At intermediate recall values, n = 30 gives better results while n = 40 gives results equivalent to regular feedback.	The performance curve for n = 20 gives consistently worse results.
5	Overall results are better than those obtained by regular feedback at desired high recalls, for all values of n tried.	Overall, for n = 62, the results are equivalent to those obtained by regular feedback.

Performance Analysis of Strategy II in
Comparison to the Regular Feedback Performance

Table 3

SERIAL NO.	COLLECTION NAME	
	MEDLARS (Fig. 9(c))	CRANFIELD
1	Initial value of m, dv rank cutoff is taken as 500, having between determined from the dv-dv rank curve (Fig. 2), such that the slope of curve at the cutoff falls below ε = 0.00004. In addition, the values of 200 and 500 for m are tried.	Calculating from the dv-dv rank curve similar to as is done for the MEDLARS collection, initial value of m = 500 is obtained. The additional value of m = 300 is used.
2	For high and intermediate recalls, m = 500 gives the best results and gives performance consistently better than that obtained by regular feedback. The increase in precision is as much as 0.08 at any recall level.	For both values of m tried, the retrieval performance is not better than that for the FDBK iteration.
3	At low recall values, m = 800 gives the best performance being within 0.04 in precision compared to the regular feedback curve.	m = 500 gives the best results approximating the regular feedback curve. For this value of m, average number of concepts in the modified fqv's is 32 compared to 114 in the regular fqv's.
4	The results are much worse for m = 200, supporting the theory that the deletion of good discriminators spoils the retreival performance.	At m = 500, the performance is worse by upto 0.04 in precision at any recall level.
5	Overall, m = 500 gives the best results being throughout better than the FDBK's performance while the average number of concepts in the modified queries is just 19 compared to 109 of FDBK queries	Overall performance is a little worse than the regular feedback performance.

Performance Analysis of Strategy III in Comparison to the Regular Feedback Performance

Table 4

SERIAL NO.	COLLECTION NAME	
	MEDLARS (Fig. 9(d))	CRANFIELD
1	The value 30 for n is used, as this gives the best performance for Strategy II. Similarly, the values 500 and 800 for m, which give the best performance for Strategy III, are used.	This experiment is not tried because observing the results obtained for the MEDLARS collection, it is expected that the performance of this strategy would be almost equivalent to Strategy III with only a very minor loss in performance.
2	The results obtained are similar to those obtained for Strategy III, being only a little worse.	
3	The best results are obtained for m = 800, n = 30, when average number of concepts in the modified fqv's is 21 as opposed to 109 in the regular fqv's.	

Performance Analysis of Strategy IV in
Comparison to the Regular Feedback Performance

Table 5

reduction is around 70% for the MEDLARS collection and around 50% for the CRANFIELD collection.

The increase in precision at any recall level is up to 0.12 for the MEDLARS collection while for the CRANFIELD collection, the performance results obtained with elongated and shortened fqv's are equivalent.

iii) Strategy III

The use of this strategy for trimming the fqv's has resulted in an improvement in performance up to 0.08 in precision at high recalls, compared to the FDBK curve for the MEDLARS collection. On the other hand, similar experiment on the CRANFIELD collection, shows a consistent loss of precision, being as much as 0.04. It should then be inferred, for the time being, that use of this strategy may give results anywhere in the range of a slight loss of performance to an appreciable improvement in performance, the benefit being that length reduction is from 70-80%.

In the case of both the collections, the initial values of dv rank cutoff, m, are chosen from the respective dv-dv rank curves by comparing the slope of the curve to the value of ε (here 0.00004). In both the cases, m, derived this way happens to be around 500 and this value of m gives the best results in both the cases. This supports the calculation of m from the dv-dv rank curve in the suggested manner.

iv) Strategy IV

As expected, the application of this strategy for reduction in the length of the fqv's results in a minor loss of performance compared to that of the previous strategy (Fig. 9(e)). Suggested values of m and n are the ones calculated for Strategy III and Strategy II respectively.

C) Comparison of the Four Strategies

The comparison of the four strategies is done in thr‹ › different ways, in the following manner:

i) P-R Curves

Fig. 9(e) shows the comparison of the best P-R curves -- one obtained for each of the four strategies, for trimming the fqv's using the MEDLARS collection. Fig. 10 shows the corresponding comparison for the CRANFIELD collection. The deductions from these figures are given below, separately for the two collections.

a) MEDLARS Collection

1) Strategy III with m = 500 is the overall best.

2) For this best strategy, at intermediate and high recalls, the precision is better than that for the regular feedback by as much as 0.08 at any recall level.

3) At very low recall level, this strategy gives a little worse performance than that for the regular feedback.

4) The average number of concepts in the shortened fqv's, using Strategy III is only 19 compared to 109 in standard feedback query vectors - a reduction of approximately 80% in the length.

5) Compared to this, Strategy I (Murray's method) with 22 as the average number of concepts in the reduced length fqv's, that is, with approximately same 80% reduction in length, gives consistently worse performance compared to the regular feedback. The precision is worse by as much as 0.06 at any recall level.

6) For approximately 70% reduction in length, even Strategy II performs better than Murray's method.

b) CRANFIELD Collection

1) Among vectors with 50% concepts chopped off, Strategy II with n = 62 and average number of concepts equal to 51 is the best along with the almost equivalently performing Strategy I with 50% concepts removed, that is with average number of concepts equal to 57 (Fig. 10).

2) With 70% reduction in the length of the vectors, Strategy I (n = 30 and the average number of concepts = 32) and Strategy III (m = 500 and the average number of concepts = 34) are almost equivalent in performance, though Strategy I has a slight edge over the latter (Fig. 10).

3) All the strategies tried give a little loss of up to 0.04 in precision but the average number of concepts is reduced by 50-70%.

ii) Overall Performance Indices

Here, four overall performance indices are compared for the best P-R curves obtained. These indices are Normalized Precision, Rank Recall and Log Precision [7]. Table 6 gives such comparison for the two collections. The figures obtained, substantiate the conclusions of the previous sub-section (i).

iii) Individual Query Behavior

a) Necessity of the Analysis

The analysis done so far has clearly established that for the MEDLARS collection, the reduced length fqv's give superior performance compared to the regular feedback with its elongated fqv's. Thus, one gets the advantages both ways -- smaller search costs because of the reduced length vectors and better retrieval performance. Moreover,

TYPE OF SEARCH PERFORMANCE MEASURE	ORIG QUERY VECTORS	1st FDBK QUERY VECTORS	REDUCED LENGTH 1ST FEEDBACK QUERY VECTORS			
			STRATEGY I	STRATEGY II	STRATEGY III	STRATEGY IV
			n=30%	n=30	m=500	m=500,n=30
NORM RECALL	0.6563	0.7646	0.7134	0.7502	0.7794	0.7594
NORM PRECISION	0.5898	0.7301	0.6312	0.7234	0.7497	0.7333
RANK RECALL	0.1399	0.2513	0.2278	0.2354	0.2335	0.2377
LOG PRECISION	0.4837	0.6165	0.5837	0.6205	0.6268	0.6272

(a) MEDLARS Collection

TYPE OF SEARCH PERFORMANCE MEASURE	ORIG QUERY VECTORS	1st FDBK QUERY VECTORS	REDUCED LENGTH 1ST FDBK QUERY VECTORS				
			STRATEGY I		STRATEGY II	STRATEGY III	STRATEGY IV
			n=30%	n=50%	n=62	m=500	
NORM RECALL	0.7496	0.8178	0.7997	0.7963	0.8216	0.7933	-
NORM PRECISION	0.6125	0.7359	0.7178	0.7180	0.7384	0.7172	-
RANK RECALL	0.2048	0.3312	0.3199	0.3249	0.3308	0.3261	
LOG PRECISION	0.4330	0.5712	0.5596	0.5637	0.5689	0.5598	-

(b) CRANFIELD Collection

Comparison of the Performance Indices for the MEDLARS and the CRANFIELD Collections

Table 6

Murray's method (Strategy I) is worse with a poorer retrieval performance.

On the other hand, the comparatively worse performance reflected by the P-R curves for the CRANFIELD collection for Strategy III is somewhat unexpected. However another surprise occurs, when one looks at the Table in Fig. 10 and finds that for the whole query collection, the total number of relevant retrieved documents among the top 10 ranks compared with those among the top 30 ranks are 405/574 for Strategy III and 392/560 for Strategy I both for approximately 70% reduction in the length of vectors, compared to 406/567 for the regular feedback. Therefore, if one considers the total number of the relevant retrieved documents as a criterion for the better performance, Strategy III really is the better of the two. The use of the total number of relevant retrieved documents as a criterion of retrieval performance is justified when one considers that from the user's point of view, the number of relevant among the retrieved documents is the more important thing; the rank of a relevant among the retrieved documents is a secondary consideration. Yet, the ranking of the relevant documents among the retrieved affects the P-R curves to quite an extent, making the true evaluation a little difficult.

This situation is further complicated by the undesirable "ranking effect" during feedback [8]. Most of the relevant documents used for positive feedback improve their ranks after the feedback iteration. This helps in improving the P-R curve which shows a better performance, even though from the user's point of view, no improvement has taken place. This is so because he has already seen these relevant documents and it does not give him any new information if these very documents improve in their rankings. We will give the name "positive ranking effect" to such a "ranking effect".

In addition, "negative ranking effect" could also occur. Suppose that after feedback, the relevant retrieved documents tha: the user has already "seen" decrease in ranks. This could happen, for instance, when relevant documents are located in two distinct regions of the document space and when one group is retrieved, the other is not. This results in a decrease in retrieval performance shown by the P-R curve, even though from the user's point of view it does not. Again, he is not concerned with whatever happens to the ranks of the documents he has already seen.

All this calls for a more detailed analysis of the individual query behavior, so that a truer picture of the actual improvement in retrieval performance can be formed.

b) CRANFIELD Collection -- Query Behavior

1) Effect of Various Concepts on Retrievability

Table 7 shows the effects of various concepts on retrievability of the individual queries for the CRANFIELD collection. From the whole query collection, three typical queries, called of Type A, B and C respectively, are chosen as examples. Using queries of Type A, the performance of Strategy III is better than that of Strategy I while the reverse is true for queries of Type B. Furthermore, queries of Types A and B show how the presence of good discriminating concepts and/or the absence of poor discriminators helps in the

QUERY TYPE	STRATEGY USED	CONCEPTS COMPRISING THE QUERY VECTOR		NO. OF RELEVANT DOCUMENTS RETRIEVED AMONG TOP 10/ TOP 30	TOTAL NO. OF RELEVANT DOCUMENTS	REMARKS
		NO. OF CONCEPTS	CONCEPT NO. (DV Rank) (Weight of each concept = 12)			
A	I	2	8341 (129), 8672 (8684)	0/0	A	i) Performance of Strategy III is better than that of Strategy I. ii) Presence of good discriminating concept number 8322(dv rank 24) helps improve the performance of Strategy III. iii) Presence of bad discriminating concept number 8672(dv rank 8684) and/or the absence of good discriminating concept number 8322(dv rank 22) makes the performance of Strategy I worse.
	III	2	8322 (24), 8341 (129)	3/3		
	FDBK	5	7042 (1653), 7261 (1222), 8322 (24), 8341 (129), 8672 (8684)	3/3		
B	I	1	8545 (59)	3/3	3	i) Performance of Strategy I is better than that of Strategy III. ii) Presence of good discriminating and single most important concept number 8545(dv rank 59) helps improve the performance of Strategy I. iii) Presence of comparatively bad discriminating and unimportant (for this query) concept number 8034 (dv rank 226) degrades the performance of Strategy III.
	III	2	8034 (226), 8545 (59)	1/3		
	FDBK	3	8034 (226), 8545 (59), 8655 (8681), 8719 (8714), 8723 (8722)	1/3		
C	I	1	8062 (796)	2/2	5	i) Sometimes concepts with high dv ranks (comparatively bad discriminators) are more important in determining the retrieval performance of a query. ii) Thus, in this query, the presence of a comparatively bad discriminating concept number 8062 (dv rank 796) is quite important and actually helps in improving the performance of Strategy I and FDBK iteration. iii) The conclusion is that the analysis based on the dv's of the concepts is not completely foolproof.
	III	1	8480 (603)	0/0		
	FDBK	2	8062 (796), 8480 (608)	2/2		

Behavior of Queries of the CRANFIELD Collection during Document Searching.

Table 7

improvement of retrieval performance. This is a pleasing result because it supports the hypothesis used as a basis for this project. However, there are comparatively very few queries of Type C, which shows that sometimes concepts with numerically high dv ranks (relatively poor discriminators) are more important in improving retrieval performance than the ones with numerically low dv ranks (relatively good discriminators). One can conclude that even though the theory of dv's is not completely foolproof, yet in the majority of cases it decides the state of affairs.

2) "Ranking Effect" and Retrievability

To determine the influence of the "ranking effects" -- both positive and negative, the documents retrieved by queries for the best case (that is, for the best parameter) of each of the strategies tried, are examined query by query for the whole of the query collection. The procedure used has been to note the ranks of the relevant documents retrieved in the ORIG iteration among the top 10 ranks (denote the set of these documents by S). As these are the documents used for positive feedback (in the present series of experiments) and thus have already been "seen" by the user, the ranks of these same documents of the set S are observed in the retrieved documents obtained for the other iterations. If the ranks of these documents have increased compared to their corresponding ranks in the retrieved documents of the ORIG iteration, during any of the regular or a modified feedback iteration, the results for this iteration suffer from the "positive ranking effect", that is the performance has been overestimated compared to what it should be.

On the other side, if the rank of any document of the set S decreases compared to its rank in the retrieved documents of the ORIG iteration, during any regular or a modified feedback iteration, the results for this iteration have the influence of the "negative ranking effect". In other words, the performance has been underestimated compared to what it should be.

Table 8 depicts the documents retrieved by two typical types of queries, called Type D and Type E, affected by such biases. For both the queries, the performance obtained for the FDBK iteration does not suffer from any appreciable ranking effect, that is, the documents of the set S basically retain the same ranks among the top 10 in the FDBK iteration as in the ORIG iteration's retrieved documents. For the query of Type D, comparatively the Strategy III retrieval results get underestimated while for the query of Type E, comparatively the Strategy I results get underestimated. Since, the Type D queries are approximately 20 in number compared to approximately 4 of Type E queries in the whole collection, it is concluded, that the Strategy III results remain somewhat underestimated compared to the Strategy I results.

c) MEDLARS Collection -- Query Behavior

The effect of the various concepts on the retrieval performance for the MEDLARS collection is found to be similar to the findings for the CRANFIELD collection.

It is, however, surprising to find that for the MEDLARS collection, the queries of Type E are absent, while there are at least 5 instances of the queries of Type D. Thus, the Strategy III results remain underestimated compared

QUERY TYPE	D			E		
STRATEGY / RANK	I	ORIG	III	I	ORIG	III
1	14R	1R	14R	270R	215	270R
2	1R	14R	1R	340	217	335
3	218R	218R	209	245	269	345R
4	258	413	207	237	345R	226
5	261	207	218R	238	266	228
6	57R	260	47R	372	265	269
7	260	151	336	227	268	227
8	282	57R	249R	226	270R	321
9	209	282	24	17	264	338
10	253	261	166R	18	247	328R
11	86R	209	261	16	369R	243
12	166R	259	259	242	246	241
13	263	166R	252R	244	371R	324
14	47R	414	251R	246	338	320
15	24	120	210	225	238	369R
16	207	24	273	267	370R	238
17	151	258	257R	228	240	325
18	91	34	85R	243	242	245
19	262	271	260	339	245	217
20	11	389	86R	268	337	333
21	252R	141	271	370R	267	313
22	259	392	250R	374	336	225
23	249R	275	374	217	374	330
24	251R	91	208	241	216	339
25	414	263	57R	344	237	323
26	34	86R	264	341	342	334
27	271	411	373	4	335	314
28	250R	296	275	345R	344	322
29	206	103	2R	373	327R	326
30	85R	281	151	371R	243	312
>30			LOST			
REMARKS	Negative "ranking effect" is more pronounced for this query using Strategy III than using Strategy I. Thus comparatively the Strategy III results get underestimated.			Negative "ranking effect" is more pronounced for this query using Strategy I than using Strategy III. Actually only positive "ranking effect" occurs using Strategy III. Thus comparatively the Strategy I results get underestimated		

Demonstration of Positive and Negative "Ranking Effects" Occurring during Document Retrieval by Feedback Queries for the CRANFIELD Collection.

Table 8

to the Strategy I results for the MEDLARS collection also. In spite of these odds, the Strategy III results have been better.

D) Overall Comparison of the Four Strategies

Comparison of the vector trimming strategies by the P-R curves shows the Strategy III to be superior for the MEDLARS collection -- it gives a performance better than the regular feedback while the average number of concepts in the modified fqv's are just 20% of the original length of the regular fqv's. Strategy I (Murray's method), on the other hand gives consistently worse performance. This is true in spite of the fact that the strategy III results get underestimated in comparison to the Strategy I results as shown above.

For the CRANFIELD collection, the performance shown by the P-R curves for Strategy III is worse than the regular feedback performance though almost equivalent to the performance for Strategy I. But, the individual query behavior and the actual total relevant retrieved documents (among the top 10 ranks / the top 30 ranks) for the whole query collection using Strategy III (405/574) compared to those for Strategy I (392/560) and the regular feedback (406/567), show the Strategy III performance to be almost equivalent to the regular feedback performance.

This shows that Strategy III may be used for reducing the lengths of the elongated fqv's with almost equivalent or better retrieval performance, while Strategy I always results in some degradation of performance compared to that of the regular feedback. Whenever vectors of fixed length are desired, Strategy II or Strategy IV may be used -- the latter being a little better.

One remarkable point about all the strategies is that a reduction in the lengths of the fqv's by even 80% maintains the performance much closer to the regular feedback (FDBK) rather than to the original (ORIG) iteration.

8. Discussion

In view of Murray's work on the construction of superior profiles [2] and in view of the present study, a few interesting questions arise concerning the initial indexing process and the application of the length reducing strategies used in conjunction with the various indexing methods. The purpose of this Section is to discuss such problems and propose some solutions.

Murray suggests using a shortened frequency ranked profile and finds that using a few broad weight categories, typically four, gives performance equivalent to that obtained by using a larger number of weight classes as in the standard weighted vectors. Even though Murray's work concerns the profile vectors, it would be expected that his results could be carried over to the indexing process for the vectors in general. Some experiments in document retrieval, using Murray's ideas, might prove the validity of this assumption, which is made in the following discussion.

A) Shortened Frequency Ranked Vectors

Murray's construction of the shortened frequency ranked vectors is performed by first forming the frequency ranked vectors and then deleting the lowest weight c ncepts. It is easy to see that Strategy III, in addition to Murray's Strategy, can also be used for reducing the lengths of the frequency ranked vectors. This is so, because the only difference between these and the standard vectors is in the formation of the c-w pairs to represent the vectors; there is no difference in the structure of the vectors.

B) A Few Categories of Weight Classes

An interesting question is: which length reducing strategy should be used when considering the vectors using only two or four weight classes? Here, using Murray's strategy of chopping off low weight concepts could wipe out some lower weight classes completely. If only one weight class is left, the resulting vector is equivalent to an unweighted vector (multiplied by a constant), which, according to Murray, could result in decreased retrieval performance. In such a case, a length reducer like Strategy III could be useful. This would, hopefully, retain all the original weight classes -- the idea being that it gives a chance of "survival" to the lower weight but good discriminating terms to show their worth in subsequent retrieval operations, e.g. during feedback.

C) Use of Negative Dictionaries

The best adjudged method for reducing the lengths of the fqv's is to retain the best discriminating concepts in each vector above an appropriately chosen cutoff m (Strategy III). Deleting the poor discriminators from each vector suggests the use of negative dictionaries. One idea that occurs is to initially index the documents and the queries utilizing Crawford's [4] negative dictionaries which use the _same_ dv rank cutoff m as the one determined for use in Strategy III. In that case, since no vector can have concepts above dv rank cutoff m at any time, there is no need for using length reducing Strategy III (or any other) at any stage of the retrieval processing. This would save computation time in addition to usual savings in storage and searching costs. The aim here is to examine the usefulness of this idea.

The above possibility is depicted by the flowchart in Fig. 11(a). On the other hand, Fig. 11(b) shows both the use of a negative dictionary with a dv rank cutoff m' at the indexing stage and the use of a length reducer with a smaller dv rank cutoff m to shorten the length of the fqv's.

In one of his experiments using the MEDLARS collection and a negative dictionary construction algorithm, Crawford has studied the effect on the retrieval performance of deleting poor discriminators from the document and the query collections. He concludes that for the MEDLARS collection, a dv rank cutoff of m' = 1000 for the negative dictionaries is the best in the sense that there is very little change in any of the performance measures for the dv rank cutoff between 1000 and 5940, while the performance decreases sharply by deleting the concepts below the dv rank cutoff of 1000.

In this project, a dv rank cutoff of m = 500 was found optimal for using Strategy III to reduce the lengths of the fqv's. It seems that for the same collection, m would be $\leq$ m'. If m = m', then using this cutoff in the negative dictionaries avoids the use of Strategy III for subsequent length reduction of the fqv's (Fig. 11(a)). If m'>m, a comparison between the performance must be obtained for the processes shown by Figs. 11(a) and 11(b). We examine the two possibilities specifically for the MEDLARS collection and take

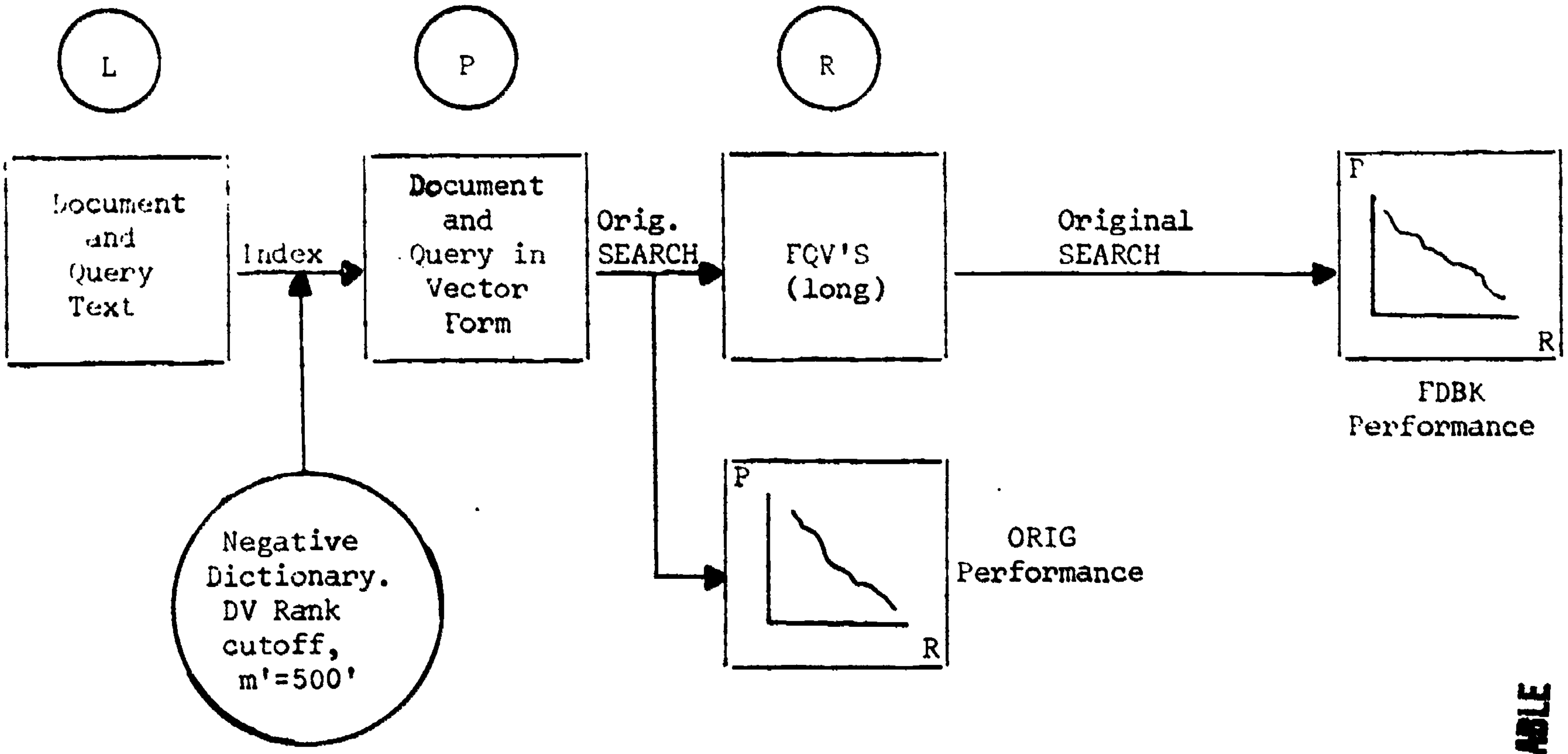


Illustration of the use of Negative Dictionaries for Relevance Feedback processing.

Fig. 11(a)

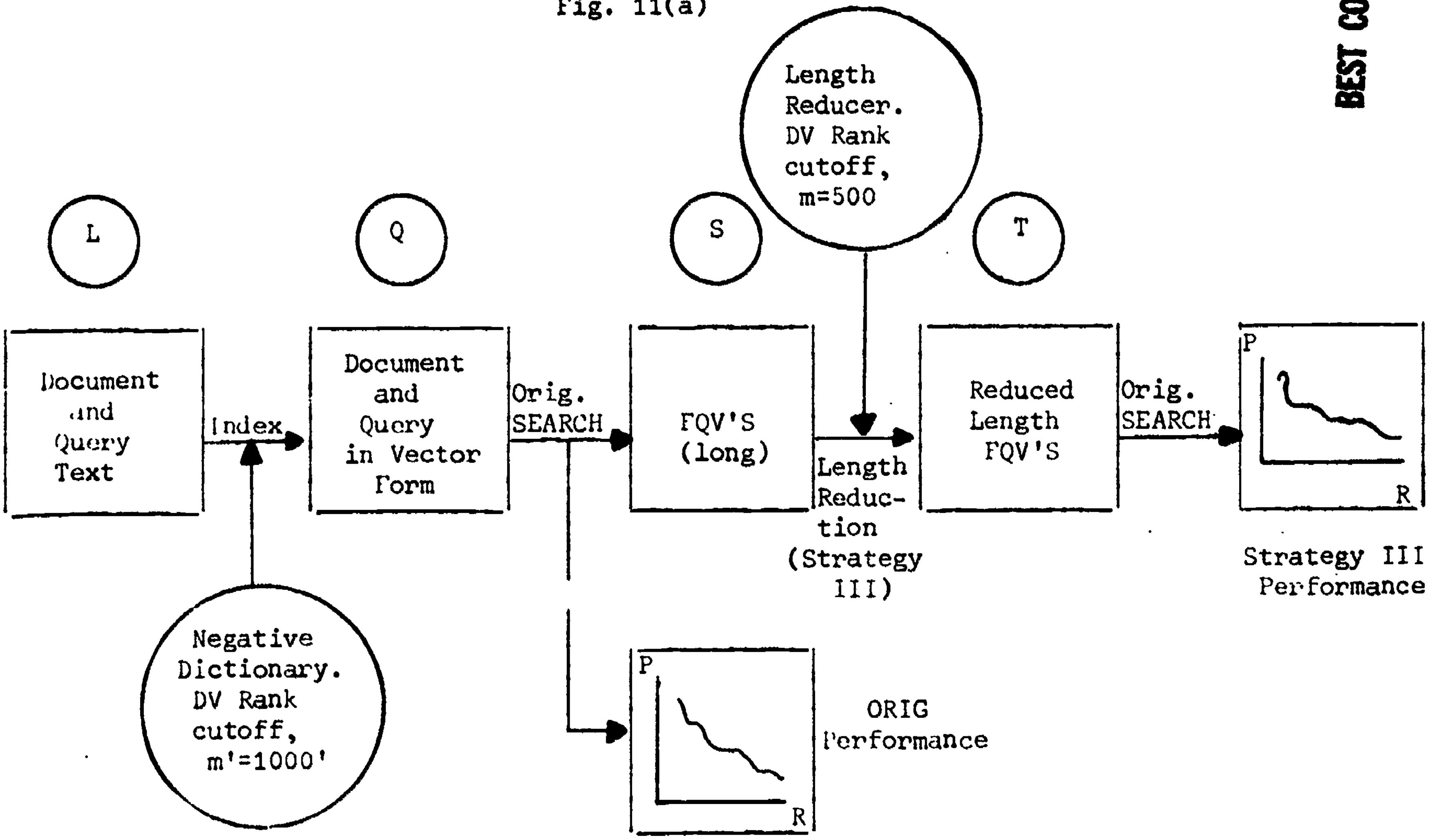

Length Reducer. DV Rank cutoff, m=500
L
Q
S
T
Document and Query Text
Index
Document and Query in Vector Form
Orig. SEARCH
FQV'S (long)
Length Reduction (Strategy III)
Reduced Length FQV'S
Orig. SEARCH
P
R
Strategy III Performance
Negative Dictionary. DV Rank cutoff, m'=1000'
P
R
ORIG Performance

Illustration of the use of both Negative Dictionaries and Length Reducers for relevance Feedback processing.

Fig. 11(b)

m = 500 and m' = 1000 based on previous experiments.

1. Crawford's results show that the retrieval performance obtained by using the dv rank cutoff of 500 in the negative dictionary (Fig. 11(a)) is appreciably worse than the retrieval performance obtained by using the corresponding cutoff of 1000 (Fig. 11(b)). This implies that after the original document search of Fig. 11(b), the relevant documents retrieved would be more in number and/or have ranks higher (numerically lower) than the corresponding relevant documents retrieved using steps in Fig. 11(a). Thus the fqv's formed by using positive feedback at stage R in Fig. 11(b) would, in general, be better formed and be composed of comparatively more concepts above dv rank of 500 (the best discriminating concepts) than in the fqv's at stage S in Fig. 11(a).

Moreover, trimming the fqv's of stage S by a length reducer employing a dv rank cutoff of m = 500 would retain all those concepts above dv rank 500 in reduced length fqv's (at stage T in Fig. 11(b)) that are present in the elongated fqv's originally. Thus the number of the best discriminating concepts (the ones above dv rank 500) in the reduced length fqv's at stage T (Fig. 11(b)) would be more than in the regular fqv's at stage R (Fig. 11(a)). Because the concepts with dv ranks above 500 are the ones that most affect the retrieval performance (Fig. 9(c)), this means that performance of the process depicted in Fig. 11(b) should be better than or at least equivalent to that of Fig. 11(a).

2. Secondly, using a smaller dv rank cutoff of 500 at the indexing stage could completely zero out some queries at stage P in Fig. 11(a), while the probability of this happening at stage Q in Fig. 11(b) after using a dv rank cutoff of 1000 in the negative dictionary is comparatively smaller. Furthermore, the query that gets wiped out at stage P in Fig. 11(a) has a lesser chance of getting zeroed out after reducing the length of the corresponding fqv at stage T in Fig. 11(b). The reason is that the fqv formed at stage S would, in general, have more concepts above dv rank 500 compared to the original query vectors of stage L.

The conclusion is in favor of using the processing steps of Fig. 11(b) rather than those of Fig. 11(a). In other words, the performance would be better by using a relaxed dv rank cutoff in the negative dictionaries at the indexing stage followed by a subsequent length reduction of the vectors using a stricter dv rank cutoff rather than using the stricter cutoff in the negative dictionaries while avoiding to use any length reduction of the vectors later on. The exact tradeoffs between the improvement in retrieval performance and savings in computation time should be further investigated experimentally.

D) Ideal Indexing

An ideal indexing would be the one in which the weight of a concept is a true indicator of the worth of the concept with respect to the particular document collection and with respect to other concepts in the same vector. An indexing method, it seems, should consider the frequencies and the distributions of all the terms in the collection. In this regard, use of the dv of a term by the indexing process is important, because dv of a term not only depends on the frequency and the distribution of that term, but also on the frequencies and the distributions of all the other terms in the collection [4].

If such an indexing scheme is used, then Strategy III for reducing the vector lengths would not be appropriate. The reason for this is that the term dv's have already been used in determining the true overall weights of the concepts in the particular context, and therefore it would seem more reasonable to chop off the low overall weight (and thus truly unimportant) terms rather than to use the dv's of the concepts again to help in reducing the vector lengths. In such a case, Murray's method (Strategy I) which eliminates the low weight concepts from elongated vectors, would be a good choice for reducing the vector lengths.

9. Summary and Conclusions

This study has made an attempt to use the dv's of the concepts to help in

a) reducing the lengths of the elongated vectors (the fqv's, in particular), and

b) the reinforcement of feedback.

Questions that arise in view of this work and the previous work in indexing, especially that by Murray [2], are examined.

The specific conclusions are:

i) Strategies for controlling the lengths of the fqv's and based on using the knowledge of the dv's of the concepts have been found quite successful.

ii) In particular, a length reducing strategy based on retaining the top best discriminating concepts only (Strategy III of this report), has been adjudged to be the most successful. This results in reducing the lengths of the vectors by 70-80%, thus saving on the storage and searching costs while at the same time attaining better or almost equal performance than that obtained with the elongated fqv's of Rocchio-type feedback.

iii) In comparison, Murray's strategy results in decreased performance than the regular feedback for the experiments conducted.

iv) When fixed length query ve..tors are desired after trimming, Strategy IV is useful and it gives only a minor loss in performance compared to the best adjudged Strategy III.

v) An interesting fact is that, the use of any strategy for 80% length reduction of the fqv's results in retrieval performance being much closer to the regular feedback with elongated vectors than to the performance of the original iteration without feedback. This means that all the strategies tried are good in the respect that they retain much of the benefits of the regular feedback with at best a comparatively little loss in performance.

vi) It is observed that Strategy III could be useful when used along with Murray's proposal of using only a few, typically four, broad categories of weight classes.

vii) It is shown that the retrieval performance is better using a relaxed dv rank cutoff in the negative dictionaries during initial indexing stage followed by a subsequent length reduction of the vectors employing a stricter dv rank cutoff rather than using the stricter cutoff in the negative dictionaries while avoiding to use any length reduction of the vectors later on.

One should note that for a large-scale application of Strategy III as a length reducer, many of the experimental details would be saved if all the concepts occurring in the document and the query collections are renumbered in their decreasing dv order.

A final remark is that the results of this project tend to support the view that the dv's and the frequencies of the concepts are important in determining the indexing process and should play a joint role in the design of an ideal indexing scheme.

References

[1] G. Salton, Editor, The SMART Retrieval System - Experiments in Automatic Document Processing, Prentice Hall, Englewood Cliffs, N.J., 1971

[2] D.M. Murray, Document Retrieval Based on Clustered Files, Ph.D. Thesis, Information Storage and Retrieval, Report ISR-20 to the National Science Foundation, Department of Computer Science, Cornell University, Ithaca, N.Y., June 1972.

[3] K. Bonwit and J. Aste-Tonsman, Negative Doctionaries, Information Storage and Retrieval, Report ISR-18 to the National Science Foundation, Section VI, Department of Computer Science, Cornell University, Ithaca, N.Y., October 1970.

[4] R. Crawford, Negative Dictionary Construction, Chapter IV, this report, Department of Computer Science, Cornell University, Ithaca, N.Y.

[5] K. Bjorklof, Relevance Feedback Based on Term Discrimination Values, Information Storage and Retrieval, Report ISR-21 to the National Library of Medicine, Section IX, Department of Computer Science, Cornell University, Ithaca, N.Y., December 1972.

[6] T. Doeppner, M. Finley and R.A. Peterson, Positive and Negative Feedback using Reinforcement Procedures Based on Term Discrimination, Information Storage and Retrieval, Report ISR-21 to the National Library of Medicine, Section X, Department of Computer Science, Cornel University, Ithaca, N.Y., December 1972.

[7] G. Salton, Automatic Information Organization and Retrieval, McGraw Hill Book Co., 1968.

[8] J.J. Rocchio, Document Retrieval Systems -- Optimization and Evaluation, Ph.D. Thesis, Information Storage and Retrieval, Report ISR-10 to the National Science Foundation, Harvard Computation Laboratory, Cambridge, Mass., March 1966.

[9] Y.K. Chang, C. Cirillo and J. Razon, "Evaluation of Feedback Retrieval using Modified Freezing, Residual Collection, and Test and Control Groups", Chapter 17, the SMART Retrieval System -- Experiments in Automatic Document Processing, Editor - G. Salton, Prentice Hall, Englewood Cliffs, N.J., 1971.

[10] H.A. Hall and N.H. 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, Ithaca, N.Y., June 1967.

The Shortening of Profiles on the Basis of Discrimination Values of Terms and Profile Space Density

Marc A. Kaplan

Abstract

The problem of long profile vectors which naturally arise in a clustered file environment is discussed. A method to reduce profile length is suggested and tested. The method involves eliminating those concept weight pairs whose concepts have poor discrimination values as defined previously by Bonwit and Aste-Tonsmann.

1. Introduction

Any information retrieval system which is to operate on a large data base and which is to provide on-line service to users must be designed to provide reasonable response time for the user.

In a SMART like [1] environment where documents and queries are represented as vectors of numbers and a query-document correlation function must be computed to evaluate the degree of similarity between each query and document one cannot hope to compute all of the correlations between a given query and every document in the collection and still give the user reasonable response time. Therefore, one seeks strategies which will reduce the number of correlations which must be computed before the retrieval system can produce an "answer".

One such strategy is to use a clustered file organization for the document space. In this organization whole groups of documents, called clusters, are represented by one pseudo-document, called the centroid or profile of the cluster. The centroid of a cluster is supposed to be representative of all the documents in its cluster. Thus the system need only correlate a query with all of the centroids in the document space and then with all of the documents under those few centroids which seem most promising rather than with each and every document in the entire collection.

One problem with clustered files is that the profile vectors are often unreasonably long, that is, the profile vectors contain many non-zero entries. Since in a usual implementation only the non-zero terms are stored, long vectors require more storage than vectors with few non-zero concept weights. (Typically vectors are stored as lists of non-zero concept-weight pairs rather than as a list of ordered weights, one for each possible concept number.

These long profile vectors occur because the profile, P, of a cluster is usually given by a formula such as:

(i) $$P = \sum_i D_i/n_i$$

where i ranges over each document in the cluster.
D_i is the i^{th} document vector
n_i is some non-zero normalizing factor

Thus the profile of a given cluster has as many different non-zero concepts as there are distinct terms in the whole cluster.

Not only do all of these non-zero terms require considerable storage but they must also certainly increase the time required to compute a correlation value. This is so since all the concept-weight pairs in the profile vector must be read into the computer and then checked against all of the concepts in the query vector. Even if none of the concepts match, the given correlation algorithm must at least search through the long profile vector to determine this.

So one must naturally ask if there is any way by which one can "shorten" the profiles of a clustered collection of documents without appreciably degrading the performance of the system with regard to recall and precision levels. If this could be achieved one would have a better retrieval system than the original system with long profiles since storage and CPU time would be conserved and the user would be served faster.

Experiments made by Murray [2] suggest that one can indeed shorten centroid vectors merely by deleting some of the concept-weight pairs. He states:

> "... profiles can be subjected to considerable deletion of low weight (less frequent) terms with little change in the quality of search output... Experiments... indicated that the deletion of 80% of the lowest weight terms drops the RC-PF only 1% to 5%."

Murray's approach is local. He examines each profile individually without considering the others and deletes those terms of lowest frequency. Might not some high frequency terms also be good subjects for deletion? Murray says:

> "On the other hand, an attempt to remove or combine related occurrences of high weight profile terms results in much poorer performance. Such procedures are to be avoided."

In this paper the experiment is based on a different approach towards finding the appropriate terms to delete from the profiles of a collection. What is desired is a technique for finding terms which are not important or may even be detrimental in computing query-centroid correlations. The experimenter does not wish to prejudge that terms should be deleted simply because they are of high, low or medium frequency. Nor does he want to guess how many terms should be deleted or kept. What then is he to do?

Bonwit and Aste-Tonsmann [3] have conducted research in the construction of what they call "negative dictionaries", that is lists of words which are best not considered as concepts in a collection of documents; which are best deleted from document vectors. One may ask whether their technique might not be applicable to the present problem?

2. Density and Discrimination

Bonwit and Aste-Tonsmann define document space density Q as:

$$\text{(ii)} \qquad Q = \frac{1}{N} \sum_{j=1}^{N} \cos(P, D_j)$$

where N = number of documents
D_j = the j^{th} document vector
P = centroid of documents given by (i) with $n_i = N$ for all i
cos is the usual cosine correlation function as used in the SMART system

Now define vector V^I as vector V with the set I of terms deleted (that is, set to zero). Then Q_I, the density of the document space with the set I of terms deleted is defined as:

(iii) $Q_I = \frac{1}{N'} \sum_k \cos(P^I, D_k^I)$

where k ranges over the set of integers for which D_k^I is not identically the zero vector

N' = number of documents for which D_k^I is non-zero (usually for small sets I, $N' = {}^kN$.)

The discrimination value, a_k, of term k is now defined to be:

(iv) $a_k = 100 * (Q_K - Q)/Q$ where $K = \{k\}$

The greater the value of a_k, the better the discrimination value of term k is said to be. Intuitively if a_k is large and positive then the deletion of term k from the document set causes t.. space to "contract", thus term k is thought to be a good term, necessary to help distinguish one document from another. If a_k is close to zero then the deletion of term k should not affect the document space significantly at all. Finally if a_k is negative then term k is a bad discriminator, its deletion will almost certainly improve the document space!

What one seeks is to find the optimum deletion set of terms, I. The hypothesis is made that that set I for which Q_I is minimized should be the "best" set of terms to delete.

In their experiments Bonwit and Aste-Tonsmann attempt to construct set I in the following way:

1. Compute a_k for each term k.

2. Construct an ordered set K, $K=(k_1,k_2,\ldots,k_t)$, such that $a_k \geq a_{k+1}$ for $i = 1,2,\ldots,t-1$, where t is the number of unique terms in the document space.

3. Let $K_p=(k_{p+1},k_{p+2},\ldots,k_t)$. Note that $K_0 = K$ and K_t = the empty set. K_p is the set of all but the best p discriminators.

4. Now one assumes that in a good deletion set, I, one should want the worst discriminators, that is one only wishes to keep the best, say p, discriminators as terms in the document space. Thus the problem of finding which of the 2^t subsets of K to use as the deletion set is reduced by the above assumption to finding which of the t subsets of the type K_p to use as a deletion set.

5. Find p to minimize Q_{K_p}. Then $I = K_p$ is the desired deletion set.

Bonwit and Aste-Tonsmann found that for the particular collection with which they worked that K_p was indeed a good deletion set. Retrieval performance as measured by normalized recall was actually improved by deleting the set of terms K_p. In fact, they found that for a sequence of document spaces, each given by deleting more and more of the "bad" discriminators, normalized recall was greatest almost at the point where Q_{K_p} was minimal.

One should notice that the deletion set K_p as computed above does not depend on any parameters external to the document space itself. There is no need to choose any frequency cutoff value nor is it necessary arbitrarily to decide the percentage of the original terms which one wishes to keep.

3. Experimental Design

The discrimination value approach is applied to the problem of long vectors by considering the set of profile vectors, apart from the rest of the collection, to be a document space. The algorithm given in part 2 is then applied to the set of profiles in order to compute the hopefully "optimum" deletion set K_p.

Existing FORTRAN programs [4] were modified by the experimenter so as to automatically compute the values Q_{K_1} for $1 = 0,1,\ldots,h$, where h equals the number of positive discriminators. The smallest value of 1 which minimized Q_{K_1} was considered to be p, the optimum number of positive discriminators to retain in the collection of profiles.

Having computed p the profile vectors were then modified by the deletion of all but the best p discriminators.

(It should be pointed out that the procedure actually used in computing the values, Q_{K_1}, is not precisely the same algorithm as given in part 2, although the results which are obtained in using the FORTRAN procedure are believed to be well described by the procedure given in section 2. The actual computations that are made are done by first computing Q in a straightforward manner, but some of the results of necessary intermediate calculations are retained. Then a simple and quick computation of the values of a_k is made by recomputing just a few intermediate values. Once the values of a_k are computed, they are sorted and a note is made of how many positive values exist. Now to compute the values, Q_{K_1}, the program first computes Q_{K_1} and saves some intermediate results.

The successive valu of $Q_{K_2}, Q_{K_3}, \ldots, Q_{K_h}$ are computed one after another by merely accounting for the effect of adding but one more term to the profile space. The interested reader is referred to the program·listing if he wishes to see the details of the computation.)

For each collection used in the experiment two sample searches were made. The first search was run with the original profile vectors. The second search was run with the modified profile vectors. Except for the use of different profile collections, the SEARCH routine of the SMART system, on which these tests were made, was given the same search parameters for each of the two runs. SEARCH results were compared using routines AVERAGE and VERIFY.

Two different collections were used in this study, both are available on disc packs on the Cornell University 360 computing system. The first, the document collection called ADIABTH DOCS was used along with its associated TREE1, the original profile collection, and QUESTS, a set of queries for which relevancy decisions have been made externally to the SMART system (82 documents, 35 queries and 13 profiles). The small size of the collection allowed for economical debugging of programs and the experimental procedure. Results obtained with this collection were encouraging enough to warrant the use of a larger collection for further study.

The second collection used was CRN4S DOCS(424 documents), QUESTS30 (30 queries), and KTREE (29 profiles). The KTREE centroid collection was created especially for this study by the experimenter, using the CLUSTER routine of the SMART system. (See computer run contained herein

```
CLUSTER - COLLECTION   CRN4S   DOCS     WILL BE CLUSTERED USING ROCCHIO'S CLUSTERING ALGORITHM

        TO FORM CLUSTER COLLECTION   CRN4S    KTREE

          THE FOLLOWING PARAMETERS WILL BE USED:

          WILL LOOSE ITEMS BE PLACED WITH THE CENTROID OF HIGHEST CORRELATION - YES
          THE NUMBER OF ITEMS TO BE FATCHED FOR CLUSTERING -     1
              (IF 0 AN OPTIMUM VALUE WILL BE CALCULATED.)
          SCRATCH UNITS   88  AND    89  WILL BE USED
          THE FIRST ITEM TO BE USED AS A POSSIBLE CLUSTER ROOT -     1
          DRPHOW - NON.POS.     PERDRP -    0.0        UNITVC - BY WORD       MINCOR -   0.800E-01      MULT -     1

THE DENSITY TEST CONTROL CARD.
          TO BE A CLUSTER POINT, AT LEAST    7 ITEMS MUST HAVE A CORRELATION GREATER THAN     0.20000
                               AND AT LEAST   15 ITEMS MUST HAVE A CORRELATION GREATER THAN     0.10000

THE CLUSTER SIZE CONTROL CARD.
          THE MINIMUM NUMBER OF ITEMS IN EACH CLUSTER IS     15
          THE MAXIMUM NUMBER OF ITEMS IN EACH CLUSTER IS     30
```

Cluster Parameters

Fig. 1

```
SEARCH - THE ADIARTH QUESTS   COLLECTION IS USED TO SEARCH AGAINST THE  ADIARTH DOCS   COLLECTION
AT LEAST  30  ITEMS WILL BE LISTED FOR EACH SEARCH
THE FIRST EIGHT CHARACTERS OF THE NAME OF EACH RESULT COLLECTION IS   ANSWER

OPTIONS FOR SEARCH :ANSWER  LTRM    : APE:

ORIG MULT   PREV MULT   MIN CORR     TYPE CORR   NORMAL     ITEMS&MULTS   CONS&WGHTS   FREEZE/FLUID   UNITVC    WGHTS DRPED   PER DRPED
    1            0      0.1000E 00   COSINE      ABNORMAL       NO             NO           FLUID       BY WORD    POS.NON.        0.0
POS MULT    NEG MULT   POS RANK CUT   NEG RANK CUT   POS CORR CUT   NEG CORR CUT   POS ATLEST   NEG ATLEST   POS NOMORE   NEG NOMORE
   0            0            0              0            1.0000         1.0000          0            0            0            0
 UNLESS     STOPALL    PREC CUTOFF     QUERIES     PKEY
   0          NO           [illegible]         SILENTAL
THERE WILL BE  1 SEARCHES MADE.  THE MAXIMUM NUMBER OF QUERIES IN EACH BATCH IS    7
```

Search Parameters

Fig. 2

```
                              P A R A M E T E R S   F O R   T R E E   S E A R C H I N G

ITERATION   0       COLLECTION      ADIABTH  LTRM                      CORRELATION    COSINE
WANTED              CORDOC            20      TIMALL            0      TIMREL             0      TIMNMR            0
GOODNESS            MCN              0.0      PCN             0.0      MLV              0.0      PLV             0.0
                    MALL             0.0      PALLOF          0.0      PALLCN           0.0      PALLLV          0.0
                    MCFCN            0.0      PFCECN          0.0      PNCFCN           0.0
                    MCNLV            0.0      PNCNLV          0.0      PVCNLV           0.0
                    MCFLV            0.0      PFCFLV          0.0      PVCFLV           0.0
SELECTION           MINNOD           1        MAXNOD          3        GAP         32767.00      EPSLON          0.01
REJECTION           MNGOOD           0.00     MNCORR          0.00     PERCOL           0.0  %   TIMWAN          0.0
PRINTING            SONS       NO             CORRELATIONS  NO         RANKS      YES
```

Parameters for Tree Searches

Fig. 3

BEST COPY AVAILABLE

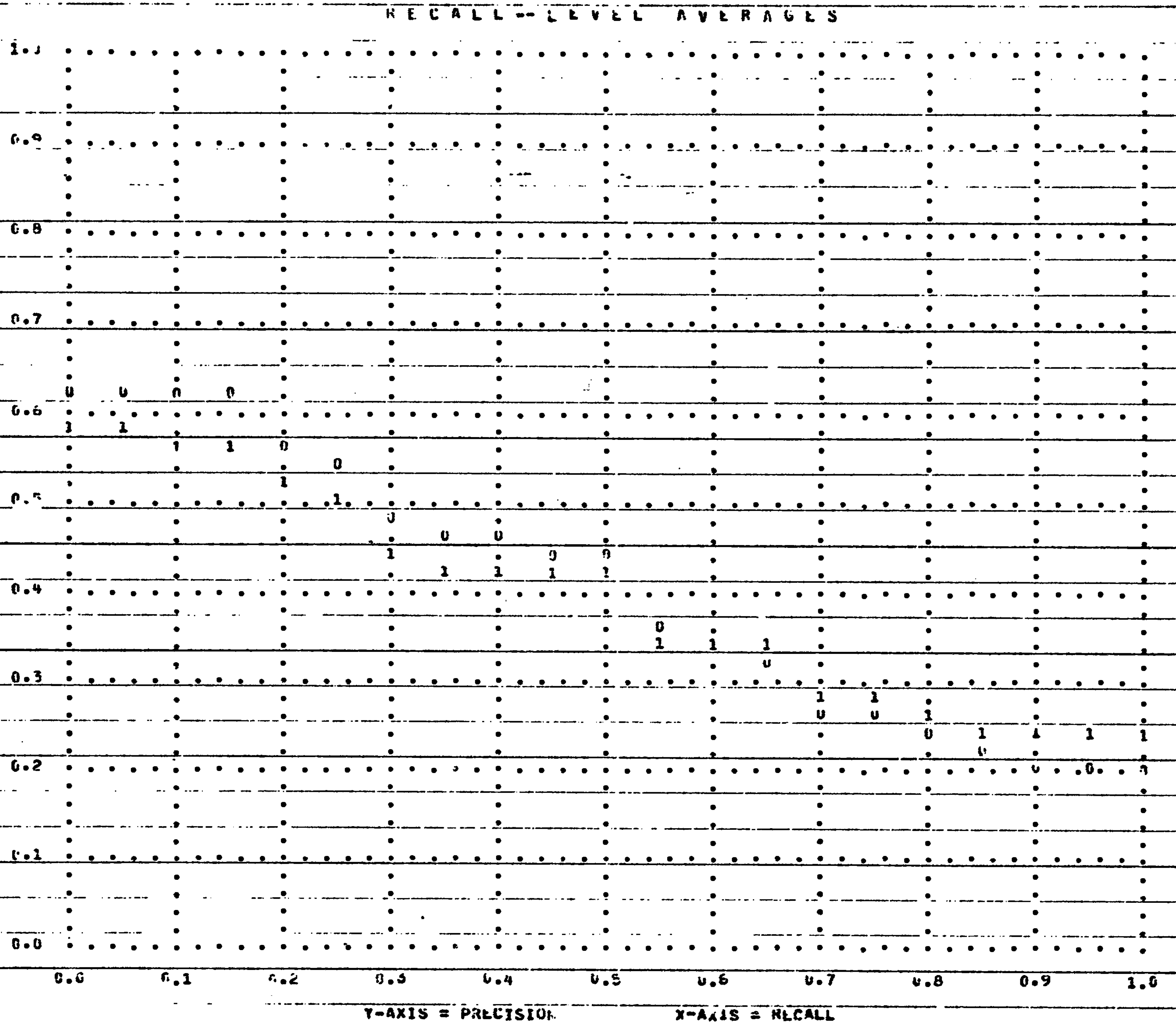
	RUN 0 Original Tree		RUN 1 Modified Tree	
RECALL	NQ	PRECISION	NQ	PRECISION
0.0	0	0.6268	0	0.5746
0.05	1	0.6268	1	0.5746
0.10	1	0.6211	1	0.5665
0.15	4	0.6125	3	0.5593
0.20	11	0.5597	12	0.5280
0.25	14	0.5328	11	0.4916
0.30	14	0.4869	11	0.4463
0.35	16	0.4526	14	0.4254
0.40	16	0.4525	14	0.4234
0.45	16	0.4468	14	0.4250
0.50	25	0.4464	15	0.4250
0.55	19	0.3523	14	0.3470
0.60	16	0.3395	13	0.3343
0.65	18	0.3286	13	0.3311
0.70	15	0.2511	11	0.2723
0.75	15	0.2511	11	0.2723
0.80	14	0.2385	11	0.2661
0.85	12	0.2122	9	0.2427
0.90	12	0.2014	9	0.2308
0.95	12	0.2014	9	0.2318
1.00	14	0.2014	11	0.2308

NORM RECALL	0.6117	0.4901
NORM PRECISION	0.5298	0.4563
RANK RECALL	0.2351	0.2550
LOG PRECISION	0.3776	0.3837

SYMBOL KEYS: NQ = NUMBER OF QUERIES USED IN THE AVERAGE
NOT DEPENDENT ON ANY EXTRAPOLATION.
NORM = NORMALIZED.

Recall Precision Results
for ADIABTH
Collection

Fig. 4



Comparison of Search Results Using Tree 1 with Modified LTRM Tree
Fig. 5

STATISTICS	T-TEST	SIGN TEST NO TIES	SIGN TEST USING TIES	WILCOXON
RANK P	0.9525	0.8805	0.9999	0.9467
LOG P	0.3753	0.8805	0.9999	0.9554
NORM P	0.5528	0.8805	0.9999	0.5629
NORM P	0.4510	0.9340	0.9999	0.5748
R-L-A .0	0.8391	0.9649	1.0000	0.8667
RLC 0.1	0.8502	0.9785	1.0000	0.8878
0.2	0.7028	0.9695	1.0000	0.7929
0.3	0.7468	0.9807	1.0000	0.8552
0.4	0.6754	0.9695	1.0000	0.8103
0.5	0.6394	0.9392	1.0000	0.7454
0.6	0.5362	0.9254	1.0000	0.6080
0.7	0.3525	0.7106	1.0000	0.3901
0.8	0.3095	0.6134	1.0000	0.3049
0.9	0.2336	0.4996	1.0000	1.2247
1.0	0.2336	0.4996	1.0000	0.2247
CUT SQU	0.9315	1.0000	1.0000	0.9785

Statistical Tests for

ADS Collection

Fig. 6

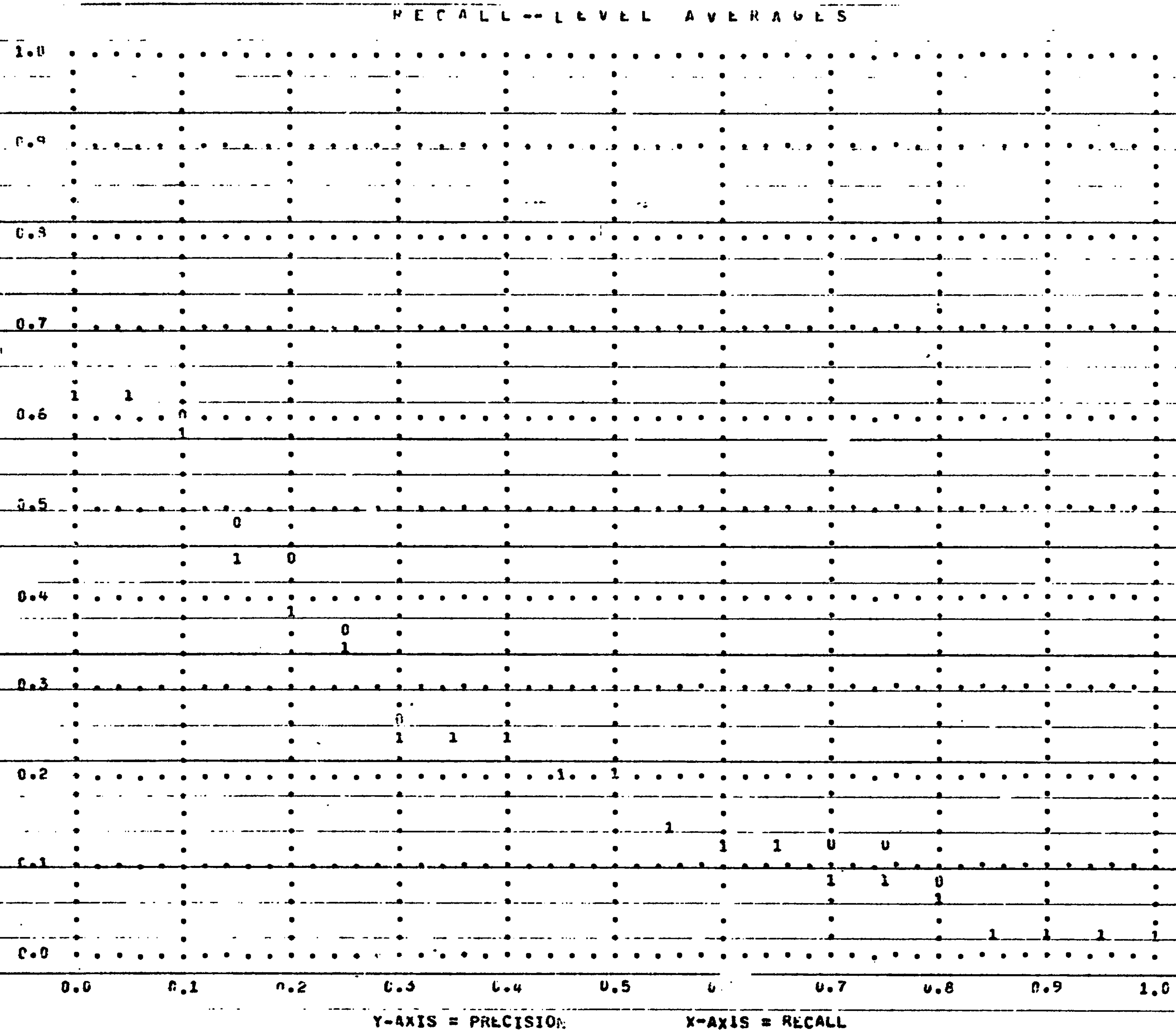
	RUN 0		RUN 1	
RECALL	NQ	PRECISION Original Tree	NQ	PRECISION Modified Tree
0.0	0	0.6222	0	0.6224
0.05	0	0.6222	0	0.6224
0.10	8	0.5904	8	0.5802
0.15	8	0.4765	6	0.4136
0.20	13	0.4376	10	0.3627
0.25	16	0.3653	16	0.3369
0.30	12	0.2699	12	0.2492
0.35	13	0.2402	14	0.2346
0.40	13	0.2402	14	0.2346
0.45	11	0.2046	12	0.2023
0.50	11	0.1933	12	0.2023
0.55	7	0.1420	7	0.1329
0.60	6	0.1276	7	0.1286
0.65	6	0.1276	7	0.1286
0.70	5	0.1200	3	0.0667
0.75	5	0.1200	3	0.0667
0.80	3	0.0707	2	0.0536
0.85	1	0.0299	0	0.0158
0.90	1	0.0299	0	0.0158
0.95	1	0.0299	0	0.0158
1.00	1	0.0299	0	0.0158

NORM RECALL	0.3638	0.3463
NORM PRECISION	0.3447	0.3260
RANK RECALL	0.0303	0.0161
LOG PRECISION	0.2765	0.2648

SYMBOL KEYS: NQ = NUMBER OF QUERIES USED IN THE AVERAGE
NOT DEPENDENT ON ANY EXTRAPOLATION.
NORM = NORMALIZED.

Recall Precision Results
for CRAN Collection

Fig. 7



Comparison of Searches Using Original and Modified XTRE Versions
Fig. 8

STATISTICS		T-TEST	SIGN TEST NO TIES	SIGN TEST USING TIES	WILCOXON
RANK R		0.7250	0.9649	1.0000	0.7478
LOG P		0.7645	0.9649	1.0000	0.8467
NORM P		0.8475	0.9649	1.0000	0.8667
NORM P		0.9020	0.9649	1.0000	0.8667
R-L-A .0		0.4958	0.6916	1.0000	0.5722
REC	0.1	0.6547	0.8895	1.0000	0.7555
	0.2	0.9594	0.9774	1.0000	0.9225
	0.3	0.7433	0.6585	1.0000	0.7555
	0.4	0.5765	0.9145	1.0000	0.6061
	0.5	0.3367	0.6585	1.0000	0.3375
	0.6	0.4816	0.6916	1.0000	0.5722
	0.7	0.9417	0.9896	1.0000	0.9695
	0.8	0.8833	0.9832	1.0000	0.9633
	0.9	0.8413	0.9774	1.0000	0.9774
	1.0	0.8413	0.9774	1.0000	0.9774
CHI SQU		0.9489	1.0000	1.0000	0.9999

Significance Tests for CRAN Collection

Fig. 9

for a description of the parameters used to create the profile collection.) This collection was used because it was the largest available collection which the experimenter could afford to use that was easily accessible on the SMART system.

4. Experimental Results

The results of the experiment are promising. As shown in Table 1 the storage requirements, in bytes, for the profile collections as maintained in the SMART system have been cut by about 30%. Also the number of concepts in the longest profile in a profile collection is cut by 33% in the case of the CRN4S collection and 47% in the case of the ADIABTH collection.

Recall and precision performance is meanwhile hardly affected at all. In the case of the CRN4S collection recall level averages are almost identical for searches using the two different centroid collections, with perhaps a slight advantage gained by using the original centroid collection instead of the modified collection. However, the results of the VERIFY routine would indicate that the difference is likely due to chance. (The overall chi-square measure was greater than .9989 for each of the three statistical tests useu: t-test, sign test, Wilcoxon test). Document level averages are also non-significantly different for the CRN4S collection. (Compıter output gives a chi-square of 1.C000 for each test.) For the ADIABTH collection examination of recall level average curves seem to show a slight trend towards the modified tree giving slightly lower precision in the low recall region of the curve and

	for CRN4S collection			ADIABTH collection		
profiles used	# of profiles	bytes required	largest # of terms	# of prof's	# of bytes	largest # terms
original	29	77836	897	13	3932	76
modified	29	52408	597	13	2792	40
savings		25428	300		1140	36
% savings		33%	33%		29%	47%

Storage economies

Table 1

higher precision in the high recall region of the curve, as compared to the results obtained with TREE1. But the significance tests show that this difference is perhaps 50% due to chance.

Glancing over the term statistics for terms thrown out and terms kept one can find terms of relatively high, low or medium frequencies for both collections which were either kept or deleted. This holds for both document frequency (number of profiles in which the term in question occurs) and overall frequency (summed weight of term throughout profile collection). Thus it appears that Murray's suggestion that high frequency terms should never be thrown is not necessarily to be followed in the future. The suggestion arose because Murray, in only looking at the profiles locally, could not tell whether or not a high frequency term was a good discriminator or a bad one, while it happens that for the most part low frequency terms are usually poor discriminators. The procedure suggested herein however, by its very definition, considers terms as they affect the profile space globally and hence can distinguish between "good" and "bad" high frequency terms.

References

[1] Gerard Salton, The SMART Retrieval System, Prentice-Hall, Inc., 1971.

[2] Daniel McClure Murray, Document Retrieval Based on Clustered Files, Information Storage and Retrieval, Report No. 20, Department of Computer Science, Cornell University, 1972.

[3] K. Bonwit and J. Aste-Tonsmann, Negative Dictionaries, Information Storage and Retrieval, Report No. 18, Department of Computer Science, Cornell University, 1970.

[4] These programs were made available to me by the SMART staff at Cornell University, as of this writing they reside in Card Files MNQ.D1, MNQ.D2, and MNQ.D3, the modified versions used in this report are included herein.

On Dynamic Document Space Modification Using Term Discrimination Values

C.S. Yang

Abstract

Brauen's algorithm for dynamic document space modification has been shown to improve retrieval effectiveness. He adds new terms to document vectors and some terms in the document vectors are increased in weight. In this study, term discrimination values are utilized so that only good terms are either added to documents or increased in weight. This can keep document vectors relatively short and poor terms are prevented from overriding good terms. The storage and retrieval effectiveness for a new version of the document space modification algorithm is studied.

1. Introduction to Dynamic Document Modification and Term Discrimination Values

In the SMART system, each document or query is represented by a vector. Document vectors may be constructed by elaborate manual work or by automatic methods. Manual construction may produce better results, but it requires too much human effort. The tendency therefore is to construct document vectors automatically by utilizing document abstracts and a dictionary, where the dictionary itself may also be constructed automatically. As a result, the well-formation of automatically constructed vectors is questioned. Besides, Brauen [1] states that "even though the vocabulary in many scientific fields is essentially standardized, it does not remain constant over time. Vocabulary in fact changes with new developments, new personnel, and other factors. As a result, document vectors, which are reasonably well defined at one time, may appear to be ill-defined five years later. Given a group of users knowledgeable in some field and given that these users submit a set of roughly similar queries, one might then expect that similar sets of documents will satisfy all these queries. A strategy designed to "group" document vectors about the user queries to which they are relevant may then aid the retrieval performance of similar queries submitted in the future."

These considerations motivate the desire to modify document vectors by user's opinions. Brauen has suggested the following algorithm for Dynamic Document Space Modification (DDSM):

1) An initial query q_o is submitted and processed. Relevance feedback iterations are performed until some modified query q_n returns a list of documents satisfactory to the user.

2) Each relevant document vector identified by the user during the feedback process is then modified as follows:
   Let D be a document relevant to q_o.

   a) If concept C_i belongs to q_o but does not belong to D, then add C_i to D with weight D^i = BETA. (1)

   b) If concept C_i belongs to both q_o and D, then modify D^i by

$$D^i = D^i + \text{GAMMA} * (120-D^i) \qquad (2)$$

c) If concept C_i belongs to D but does not belong to q_o, then modify D_i by

$$D_i = D_i - (\frac{D_i}{DELTA} + 1) \qquad (3)$$

Brauen has shown that this algorithm does improve retrieval performance because documents tend to center around those queries to which they are relevant. [1] But two questions need to be considered:

1) From equation 1, new terms are added to the document vectors. After the document space is modified by many queries, is it possible that so many new terms are added to the document vectors that the vectors are unreasonably long? If so, document storage becomes a serious problem. On the other hand, the COSINE correlation between two vectors is a term matching process. It takes more time to calculate the correlation between longer vectors.

2) Are all terms in the original queries good ones? If this is not necessarily the case, is it wise to judge the usefulness of a term which is in the query but not in the document before the term is added to the document vector? Similarly, is it wise to judge the usefulness of a term which is in both the query and the document before it is increased in weight?

With respect to the first question, several observations are made based on experiments using the CRN4S DOCS document space (424 documents) and CRN4S QUESTS query space (155 queries). The results of the earlier experiments show that many documents are significantly increased in length. A typical example is the following:

Query 4 has 12 concept terms and document 18 has 76. Query 4 retrieves document 18 as a relevant document with rank 6. Eight terms in Query 4 but not in document 18 are added to document 18. Later on, Query 5 also retrieves document 18 with rank 6 as a relevant document. Seven terms are added to this document. Document 18 thus increases from 76 terms to 91 tarms after being modified by only two queries.

When a document vector has been lengthened to a certain extent is it possible that relatively few new terms will be added to it? To obtain an estimation of this, consider how the longest vector (document 234) in CRN4S DOCS behaves. This document has 186 terms. Query 23 retrieves it with rank 5. It is found that 6 out of the 8 terms in query 23 are not in document 234. If a long vector is lengthened so quickly, an originally short vector (short vectors in CRN4S DOCS have lengths of about thirty terms) will probably double or triple in length after being modified by many queries.

The above arguments support the desirability of limiting document lengths. A natural way to do this is to modify Brauen's algorithm so that only good terms are added to documents (by equation 1) and poor terms are deleted from document vectors (by equation 2).

The "goodness" of a term was first studied by Bonwit and Aste-Tonsmann [2]. A term is considered to be a good term, i.e., a discriminator, if its distribution is such that it serves to distinguish or discriminate among documents in the collection. Otherwise it is a non-discriminator. For example, if a term occurs in almost all documents in a collection, it may not be a good discriminator since it may have little effect in distinguishing among the documents.

A formulation of a term discrimination value function developed by Crawford [3] is introduced as follows:

Let $D_1, \ldots, D_N$ be a collection of N documents. Each document, D_i, is represented by a vector $D_i = (D_{i1}, D_{i2}, \ldots, D_{iM})$ where M is the number of terms in the dictionary for the collection and D_{ij} is the weight of concept j in document i.

The centroid vector, $C = (C_1, \ldots, C_M)$, of the document collection is defined as

$$C_j = \frac{1}{N} \sum_{i=1}^{N} D_{ij} \qquad j = 1, 2, \ldots, M$$

The Compactness, Q, of the document collection is

$$Q = \frac{1}{N} \sum_{i=1}^{N} \cos(C, D_i), \qquad 0 < Q \leq 1.$$

The Compactness of the document collection with term i deleted is

$$Q_i = \frac{1}{N} \sum_{j=1}^{N} \cos(C^i, D_j^i)$$

where C^i and D_j^i are respectively the centroid vector and jth document vector with term i deleted. The **Discrimination** Value, V_i, of term i is defined as

$$V_i = \frac{Q_i - Q}{Q} * 100$$

Since a good term can distinguish among the documents, its existence makes the document space more sparse, i.e., less compact.

Then $Q_i > Q$ for a good discriminator C_i.

Conversely, a poor term fails to distinguish among the documents, its existence makes the space more compact. Then $Q_i < Q$ for a poor discriminator C_i.

So, the higher the discrimination value for a term, the better discriminator it is. If $V_i < o$ for term C_i, C_i is called a non-discriminator.

The above theory therefore generates a coefficient proportional to the usefulness of a term.

2. Dynamic Document Space Modification Using Term Discrimination Values

The following modified version of Brauen's algorithm is proposed to alleviate the problems mentioned before.

1) An initial query q_o is submitted and processed. Relevance feedback iterations are performed until some modified query q_n returns a list of documents satisfactory to the user.

2) Each relevant document vector identified by the user during the feedback process is then modified as follows: let D be a document relevant to q_o .

   a) Concept C_i belongs to q_o but does not belong to D. If C_i is a good discriminator then add C_i to D with weight D^i = BETA.

   b) Concept C_i belongs to q_o and D. If C_i is a good term then modify D^i by

   $$D^i = D^i + \text{GAMMA} * (120 - D^i)$$

   c) Concept C_i belongs to D but does not belong to q_o. Modify D^i by

   $$D^i = D^i - (\frac{D_i}{\text{DELTA}} + 1).$$

Poor discriminators in the queries are not added to document vectors by the extra condition in (a). Also, poor terms in the documents do not gain weight by the condition in (b). From the point of view of performance, if a document space is full of poor terms with high weights, they will override the effect of good terms and hence the retrieval effectiveness will deteriorate.

3. Experiment

The following data bases are used:

Document Space:	CRN4S DOCS (424 documents)
Query Space:	CRN4S QUESTS (155 queries)
Tree Structure:	REW-CRN4S TREE (63 first level centroids, 14 second level centroids, 1 root node)

Brauen calls two queries similar if they have three or more terms in common. Otherwise two queries are nonsimilar. He divides the 155 queries into two subsets. 125 of them are used to modify the document space and are called the Modification Set. Among the remainder of the thirty queries, fifteen are similar to some queries in the Modification Set and fifteen are not similar to any query in the Modification Set. The thirty queries are used to test the effectiveness of the document collection modified by the 125 queries. His convention is adopted in this experiment. Both sets of queries are shown in Table 1.

A new load module called SMARTDSM has been set up to accommodate the document space modification and the Retirement policy. The latter is not discussed in this paper. One can choose

Query Number	Cranfield Query Number	Number of Similar Queries	Similar Queries
1	12	0	-
2	15	0	-
3	48	0	-
4	66	0	-
5	72	0	-
6	87	0	-
7	90	0	-
8	96	0	-
9	100	0	-
10	104	0	-
11	106	0	-
12	117	0	-
13	119	0	-
14	121	0	-
15	124	0	-
16	6	3	24,33,56
17	9	3	4,5,111
18	24	7	6,23,29,33,49,56,147
19	27	3	28,47,58
20	30	6	13,28,109,110,113,150
21	33	8	6,24,34,36,56,63,77,86
22	39	7	13,28,37,41,58,150,151
23	51	3	46,50,118
24	57	4	14,56,83,115
25	63	5	33,109,110,147,148
26	78	3	1,83,135
27	81	6	82,94,95,136,143,144
28	146	3	133,134,145
29	147	5	23,24,49,63,148
30	148	3	49,63,147

Summary of Information
Test Set Queries

Table 1

to run DDSM (with or without term discrimination considerations) and/or document retirement by properly setting several parameters.

Crawford's program calculates the discrimination value of each of the 4439 terms in the dictionary for the Cranfield 424 document collection. All terms are ordered and renumbered in descending discrimination value so that concept 1 has the highest discrimination value and concept 4439 has the lowest discrimination value. The document-, query-, and tree-structure vectors are recoded according to this new dictionary. In this new environment, the modified Brauen method can be restated as follows:

1) An initial query q_o is submitted and processed. Relevance feedback iterations are performed until some modified query q_n returns a list of documents satisfactory to the user.

2) Each relevant document vector identified by the user during the feedback process is modified as follows: Let D be a document relevant to q_o.

   a) Concept C_i belongs to q_o but does not belong to D. If $C_i <$ GTERM then add C_i to D with weight $D^i = \text{BETA} * (1 + \frac{1}{2} * \text{ONE})$. If $\text{GTERM} \leq C_i < \text{NTERM}$ then add C_i to D with weight $D^i = \text{BETA}$.

   b) Concept C_i belongs to q_o and D. If $C_i <$ NTERM then modify D^i by $D^i = D^i + \text{GAMMA} * (120 - D^i)$.

   c) Concept C_i belongs to D but does not belong to q_o. Modify D^i by

$$D^i = D^i - (\frac{D^i}{\text{DELTA}} + 1).$$

GAMMA = 0.225 and DELTA = 8 are the optimal values obtained by Brauen and are used throughout this study.

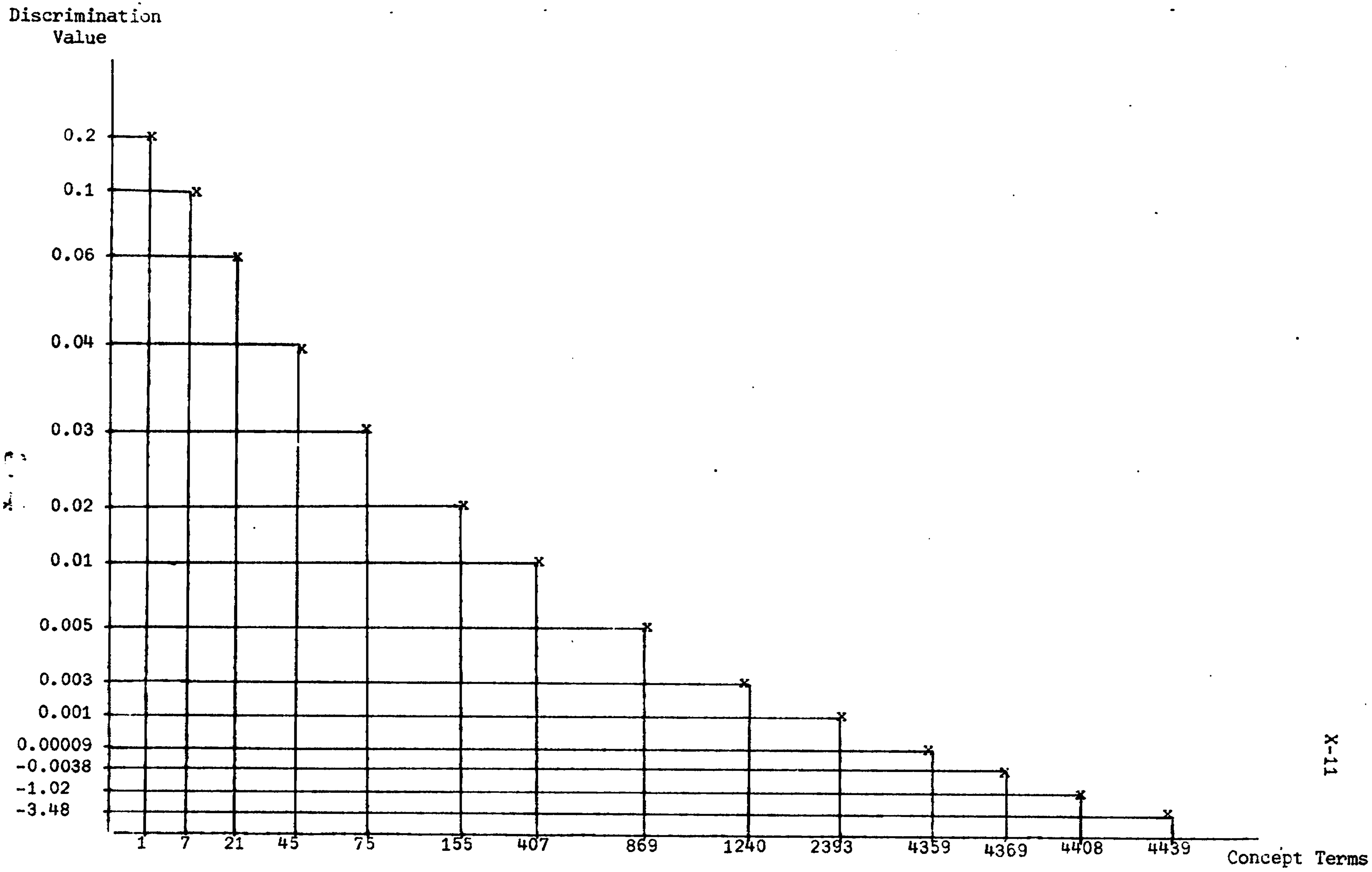
In (a), the parameter ONE can be set to 1 or 0. If ONE = 1, more emphasis is put on the very high discrimination value terms added to documents. If ONE = 0, all terms added are treated equally. If ONE = 1 and BETA = 20, terms with very high discrimination values are added with normal weight 30 and other terms added have weight 20.

The distribution of discrimination values of the 4439 terms is shown in Fig. 1. The first 407 terms in the dictionary have discrimination values greater than 0.01. They are considered to be very good terms. The last 80 terms have negative discrimination values. Their document frequencies range from 46 to 219. These are therefore very high frequency terms.

Table 2 shows the seven sets of GTERM and NTERM cutoffs used in this experiment. The 125 queries modify the document space for each set of cutoffs. Two iterations are run. Relevant documents in each iteration with rank above 30 are modified. If every query modifies six documents on the average, there are 6x125=750 modifications, and each document is modified 750/424 ≈ 1.77 times. The remaining thirty queries then test the retrieval performance of the resulting document spaces. The number of terms added to the document space is also counted.

4. Discussion of Results

Table 2 summarizes the experimental conditions. No document space modification is performed for set 0. This is a standard search run used for comparison.



Term **discrimination** values for the 4439 terms in the dictionary for CRN4S DOCS. X-Y axes are not drawn according to scale. (This curve is intended to give an idea of the distribution of discrimination values for the 4439 terms.)

Fig. 1

SET	GTERM	NTERM	BETA	ONE	NEW TERMS ADDFD	AVERAGE DOC. LENGTH INCREASE	COMMENT
0	-	-	-	-	0	0	Standard search No DDSM
1	0	4439	30	0	2969	7	Standard DDSM
2	407	4360	30	1	2403	5.7	DDSM with Term Discrimination considered
3	407	2465	30	1	2287	5.4	DDSM with Term Discrimination considered
4	407	2465	20	1	2282	5.4	DDSM with Term Discrimination considered
5	407	1000	30	1	1834	4.3	DDSM with Term Discrimination considered
6	155	600	30	1	1441	3.4	DDSM with Term Discrimination considered

Seven sets of parameters used in the experiment with or without DDSM and Term Discrimination consideration

Table 2

There is no cutoff in the DDSM for Set 1. This is the standard Brauen method. 2969 terms are added to the 424 documents. Each document is lengthened by 7 terms on the average.

For Set 2, only the eighty negative discrimination value terms are deleted. Because of the high frequencies of the 80 terms, the number of terms added to the collection is decreased quickly from 2969 to 2403.

Set 6 exhibits the largest cutoff. Compared with the standard DDSM (Set 1) where 2969 terms are added, the number of terms added is decreased by $(\frac{2969-1441}{2969}) \times 100\% = 51.5\%$.

Sets 3 and 4 have intermmediate cutoff values and $(\frac{2969-2287}{2969}) \times 100\% = 23\%$ of the terms are prevented from being added to the document collection.

The seven sets are tested with the similar and nonsimilar test sets respectively. Their retrieval performances are shown in Tables 3 to 6 and plotted in Fig. 2 to 7. The performances can be summarized as follows:

1) Similar queries in the oth iteration.
The original Brauen's DDSM (Set 1) shows a considerable superiority over all others. Sets 2, 3, 4, 5, and 6 are almost the same at all recall levels. They have poorer precision than that of Set 0 at very low recall levels but are universally better at recall levels above 0.3. The significance tests show that the superiority of any one of the five sets (Sets 2, 3, 4, 5, 6) to the other four sets is not significant. One may conclude that these five sets have approximately the same performance

	Set 0		Set 1		Set 2		Set 3		Set 4		Set 5		Set 6	
RECALL	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION
0.0	0	0.7265	0	0.7169	0	0.5917	0	0.5917	0	0.5869	0	0.6194	0	0.6291
0.10	4	0.6931	4	0.6947	4	0.5917	4	0.5917	4	0.5758	4	0.6194	4	0.5958
0.20	9	0.6216	9	0.6947	9	0.5540	9	0.5540	9	0.5469	9	0.4884	9	0.5799
0.30	11	0.5083	11	0.6541	11	0.4775	11	0.4775	11	0.4509	11	0.4553	11	0.5114
0.40	14	0.2887	14	0.5540	14	0.4222	14	0.4222	14	0.4111	14	0.3922	14	0.4158
0.50	14	0.2869	14	0.4644	14	0.3793	14	0.3827	14	0.3738	14	0.3767	14	0.3913
0.60	14	0.2029	14	0.4137	14	0.2776	14	0.2853	14	0.2828	14	0.2778	14	0.2857
0.70	13	0.1717	13	0.3330	13	0.2320	13	0.2353	13	0.2259	13	0.2419	13	0.2391
0.80	12	0.1527	12	0.2810	12	0.2071	12	0.2107	12	0.2050	12	0.2183	12	0.2221
0.90	12	0.1252	12	0.2150	12	0.1634	12	0.1628	12	0.1633	12	0.1749	12	0.1704
1.00	12	0.1157	12	0.2106	12	0.1615	12	0.1609	12	0.1621	12	0.1619	12	0.1692
NORM RECALL		0.8499		0.8652		0.8563		0.8564		0.8545		0.8549		0.8567
NORM PRECISION		0.6381		0.7115		0.6600		0.6605		0.6526		0.6544		0.6661
RANK RECALL		0.1632		0.2933		0.2130		0.2154		0.2135		0.2197		0.2225
LOG PRECISION		0.4246		0.5235		0.4578		0.4594		0.4551		0.4573		0.4661

SYMBOL KEYS: NQ = NUMBER OF QUERIES USED IN THE AVERAGE NOT DEPENDENT ON ANY EXTRAPOLATION.

NORM = NORMALIZED.

Test Results for Similar Test Set at 0th Iteration

Table 3

	Set 0		Set 1		Set 2		Set 3		Set 4		Set 5		Set 6	
RECALL	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION
0.0	0	0.8734	0	0.9340	0	0.9340	0	0.9340	0	0.9340	0	0.9340	0	0.9340
0.10	4	0.8734	4	0.9340	4	0.9340	4	0.9340	4	0.9340	4	0.9340	4	0.9340
0.20	9	0.8467	9	0.9340	9	0.9340	9	0.9340	9	0.9340	9	0.9340	9	0.9340
0.30	11	0.7824	11	0.9340	11	0.9073	11	0.9073	11	0.9340	11	0.9340	11	0.9340
0.40	14	0.5865	14	0.8293	14	0.8083	14	0.8083	14	0.8490	14	0.8153	14	0.8562
0.50	14	0.5827	14	0.7875	14	0.7867	14	0.7867	14	0.7990	14	0.7637	14	0.8131
0.60	14	0.5369	14	0.7001	14	0.7150	14	0.7152	14	0.6889	14	0.6551	14	0.6959
0.70	14	0.3607	14	0.4835	14	0.5046	14	0.5051	14	0.4727	14	0.4654	14	0.4710
0.80	14	0.3365	14	0.4487	14	0.4368	14	0.4372	14	0.4271	14	0.4184	14	0.4244
0.90	14	0.2722	14	0.3897	14	0.3811	14	0.3819	14	0.3540	14	0.3600	14	0.3557
1.00	14	0.2696	14	0.3832	14	0.3770	14	0.3777	14	0.3483	15	0.3494	14	0.3458
NORM RECALL		0.8950		0.9050		0.9078		0.9078		0.9087		0.9066		0.9091
NORM PRECISION		0.7574		0.8252		0.8271		0.8271		0.8275		0.8233		0.8305
RANK RECALL		0.3388		0.4566		0.4556		0.4563		0.4374		0.4356		0.4402
LOG PRECISION		0.5925		0.6790		0.6796		0.6797		0.6739		0.6693		0.6774

SYMBOL KEYS: NQ = NUMBER OF QUERIES USED IN THE AVERAGE NOT DEPENDENT ON ANY EXTRAPOLATION.
NORM = NORMALIZED.

Test Results for Similar Test Set at First Iteration

Table 4

	Set 0		Set 1		Set 2		Set 3		Set 4		Set 5		Set 6	
RECALL	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION
0.0	0	0.7086	0	0.7338	0	0.7946	0	0.7942	0	0.7435	0	0.7929	0	0.7385
0.10	2	0.7086	2	0.7264	2	0.7946	2	0.7942	2	0.7361	2	0.7929	2	0.7290
0.20	14	0.6370	14	0.6653	14	0.6720	14	0.6715	14	0.6578	14	0.6545	14	0.6660
0.30	15	0.4998	15	0.5650	15	0.5554	15	0.5549	15	0.5443	15	0.4932	15	0.5507
0.40	15	0.4576	15	0.5088	15	0.5079	15	0.5074	15	0.4993	15	0.4622	15	0.5199
0.50	15	0.4515	15	0.4997	15	0.5009	15	0.5003	15	0.4614	15	0.4552	15	0.5123
0.60	15	0.3781	15	0.4837	15	0.4738	15	0.4715	15	0.4256	15	0.4237	15	0.4567
0.70	15	0.3239	15	0.4783	15	0.4632	15	0.4564	15	0.3688	15	0.4081	15	0.4332
0.80	13	0.2859	14	0.4438	14	0.4386	14	0.4355	14	0.3341	14	0.3976	14	0.3980
0.90	7	0.1795	9	0.2559	9	0.2342	9	0.2303	8	0.2157	8	0.2156	9	0.2268
1.00	7	0.1750	9	0.2554	9	0.2326	9	0.2274	8	0.2097	8	0.2156	9	0.2252
NORM RECALL		0.8575		0.8835		0.8866		0.8863		0.8763		0.8775		0.8856
NORM PRECISTON		0.6892		0.7262		0.7368		0.7355		0.7117		0.7192		0.7260
RANK RECALL		0.2256		0.3251		0.3091		0.3068		0.2667		0.2734		0.2952
LOG PRECISION		0.4979		0.5622		0.5616		0.5602		0.5254		0.5337		0.5487

SYMBOL KEYS: NQ = NUMBER OF QUERIES USED IN THE AVERAGE
NOT DEPENDENT ON ANY EXTRAPOLATION.
NORM = NORMALIZED.

Test Results for Nonsimilar Test Set at 0th Iteration

Table 5

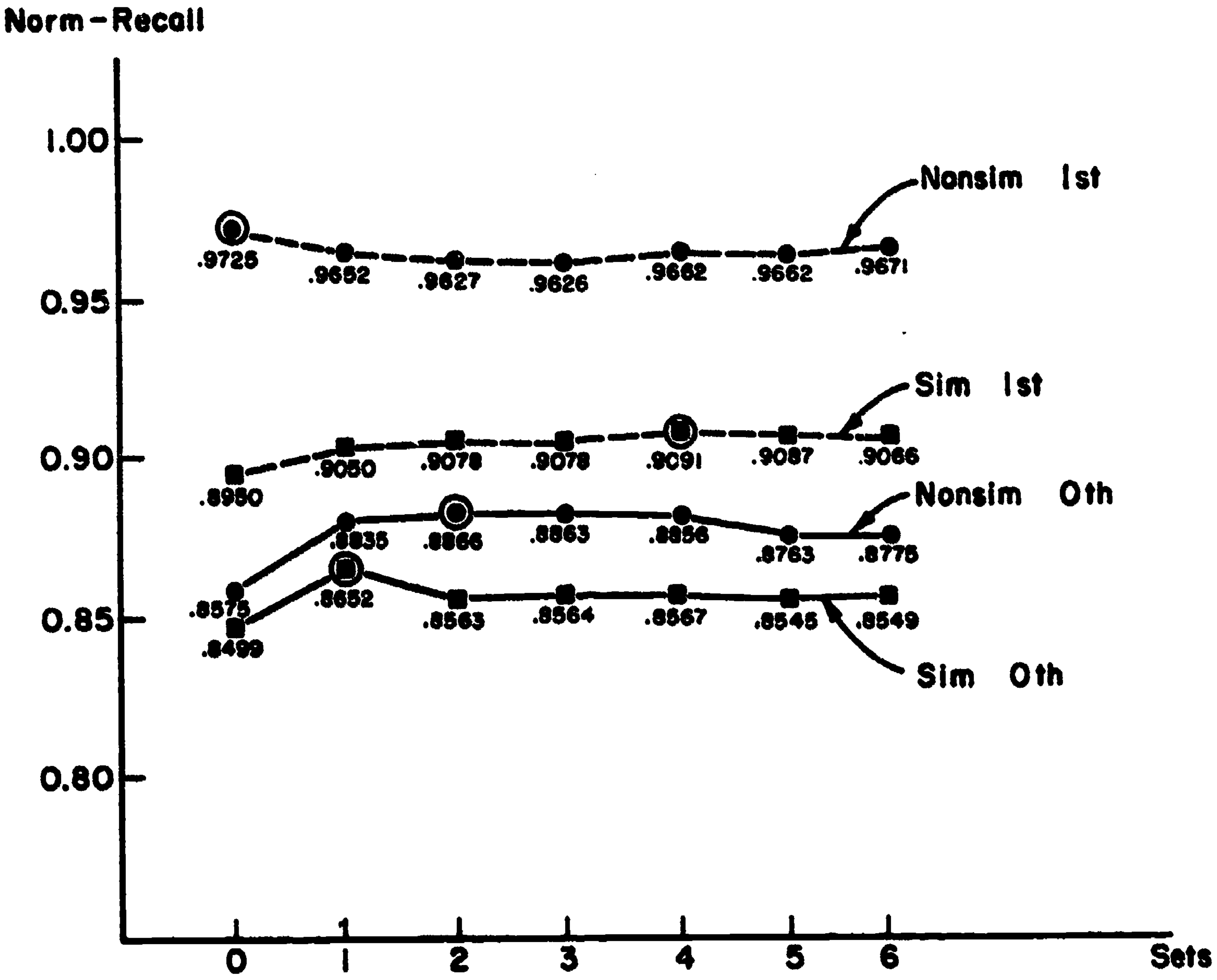
	Set 0		Set 1		Set 2		Set 3		Set 4		Set 5		Set 6	
RECALL	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION	NQ	PRECISION
0.0	0	1.0000	0	0.9431	0	1.0000	0	1.0000	0	1.0000	0	0.9429	0	1.0000
0.10	2	1.0000	2	0.9431	2	1.0000	2	1.0000	2	1.0000	2	0.9429	2	1.0000
0.20	14	0.9181	14	0.9431	14	1.0000	14	1.0000	14	0.9733	14	0.9429	14	1.0000
0.30	15	0.8781	15	0.7939	15	0.8678	15	0.8678	15	0.8675	15	0.8929	15	0.8669
0.40	15	0.8587	15	0.7606	15	0.8437	15	0.8342	15	0.8309	15	0.8706	15	0.8325
0.50	15	0.8065	15	0.7426	15	0.8426	15	0.8331	15	0.8309	15	0.8319	15	0.8325
0.60	15	0.6228	15	0.6788	15	0.7450	15	0.7346	15	0.7246	15	0.7417	15	0.7260
0.70	15	0.5380	15	0.6634	15	0.6579	15	0.6571	15	0.6376	15	0.6580	15	0.6426
0.80	15	0.5050	15	0.6156	14	0.6176	14	0.6172	14	0.5938	15	0.5936	14	0.6192
0.90	15	0.3887	14	0.4465	14	0.3899	14	0.3955	14	0.4461	14	0.3772	14	0.4397
1.00	15	0.3767	14	0.4465	14	0.3899	14	0.3955	14	0.4461	14	0.3721	14	0.4397
NORM RECALL		0.9725		0.9652		0.9627		0.9626		0.9662		0.9671		0.9662
NORM PRECISION		0.8712		0.8635		0.8889		0.8879		0.8903		0.8818		0.8921
RANK RECALL		0.4856		0.5167		0.4871		0.4843		0.5286		0.4671		0.5278
LOG PRECISION		0.7092		0.7216		0.7354		0.7320		0.7369		0.7218		0.7391

SYMBOL KEYS: NQ = NUMBER OF QUERIES USED IN THE AVERAGE NOT DEPENDENT ON ANY EXTRAPOLATION.

NORM = NORMALIZED.

Test Results for Nonsimilar Test Set at First Iteration

Table 6



Norm-Recall curves. Circled point is the maximum point in the curve.

Fig. 2

SIMILAR TEST SET

■————■ oth iteration

■– – – – – ■ 1st iteration

NONSIMILAR TEST SET

●————● oth iteration

●– – – – – ● 1st iteration

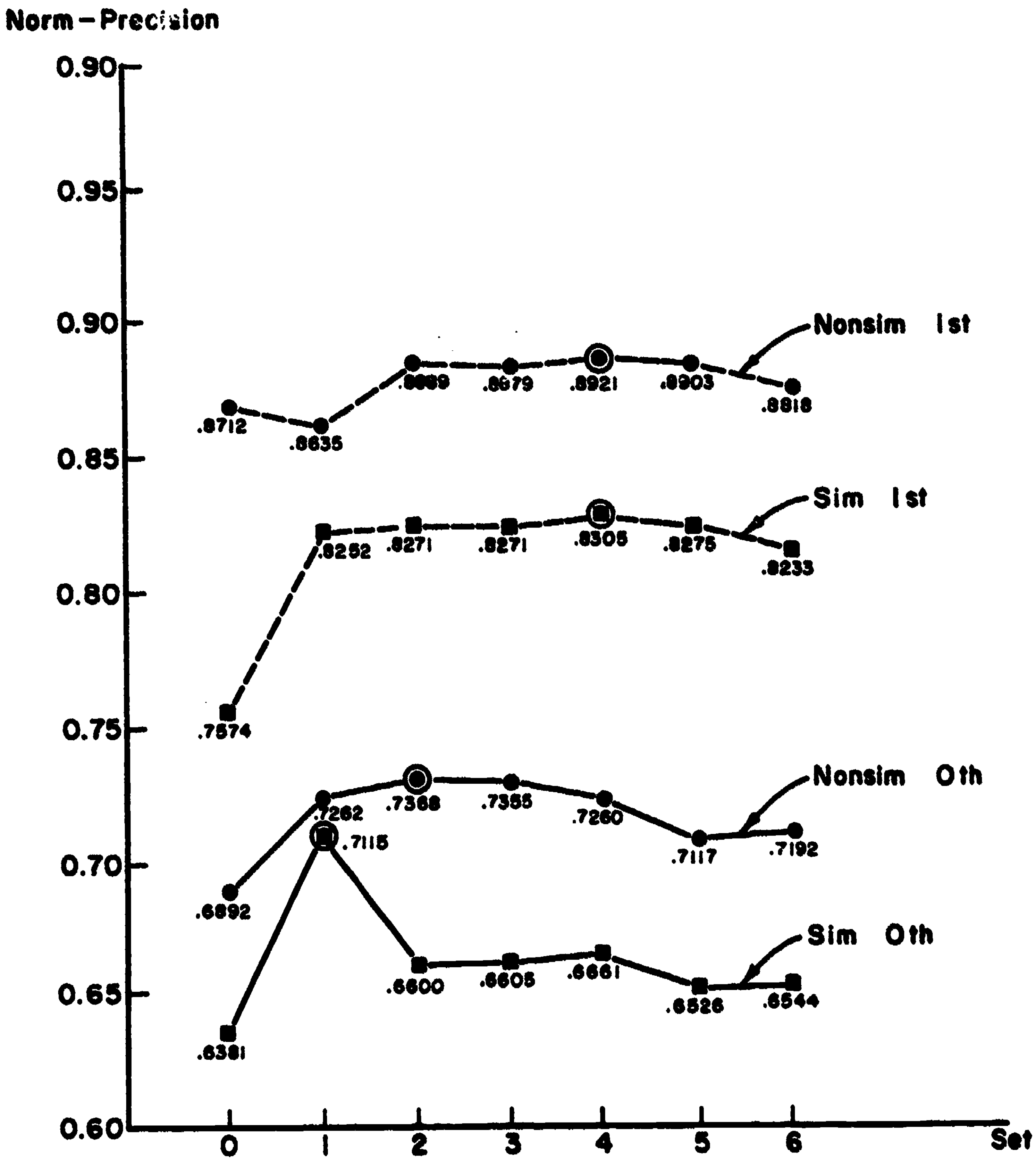


Fig. 3. Norm-Precision Curves. Circled point is the maximum point in the curve.

SIMILAR TEST SET

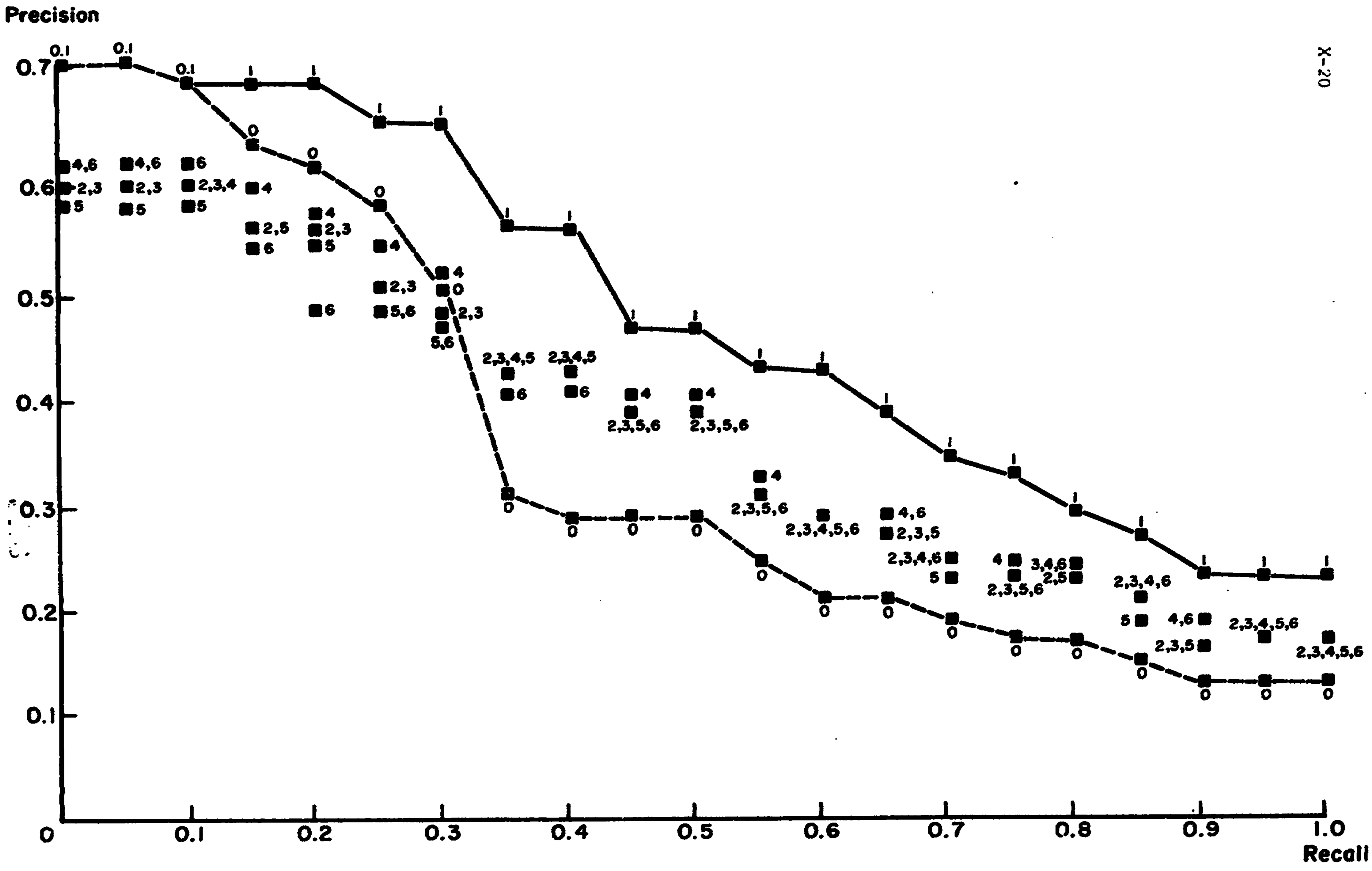
■————■ oth iteration

■– – – – – ■ 1st iteration

NONSIMILAR TEST SET

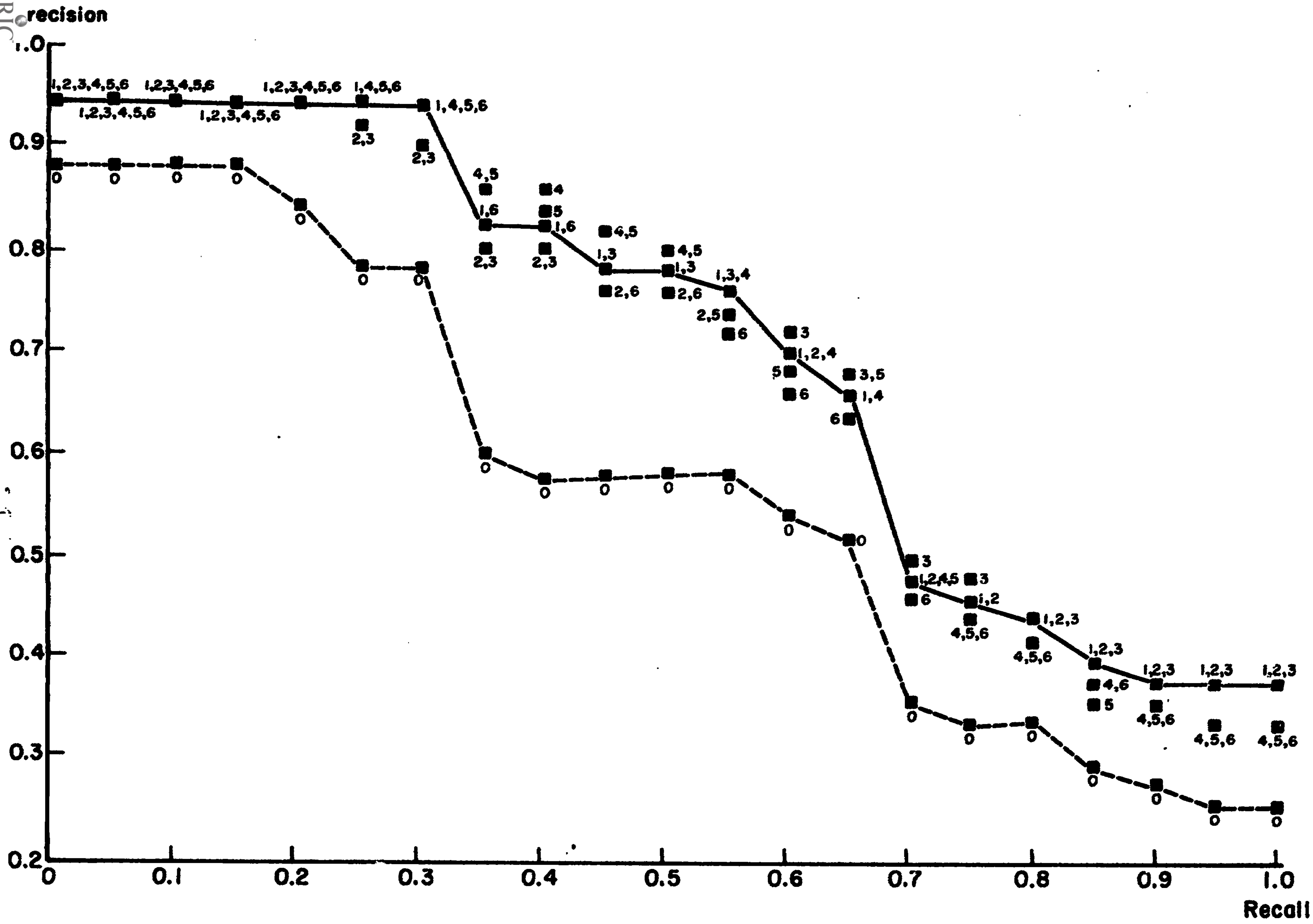
●————● oth iteration

●– – – – – ● 1st iteration



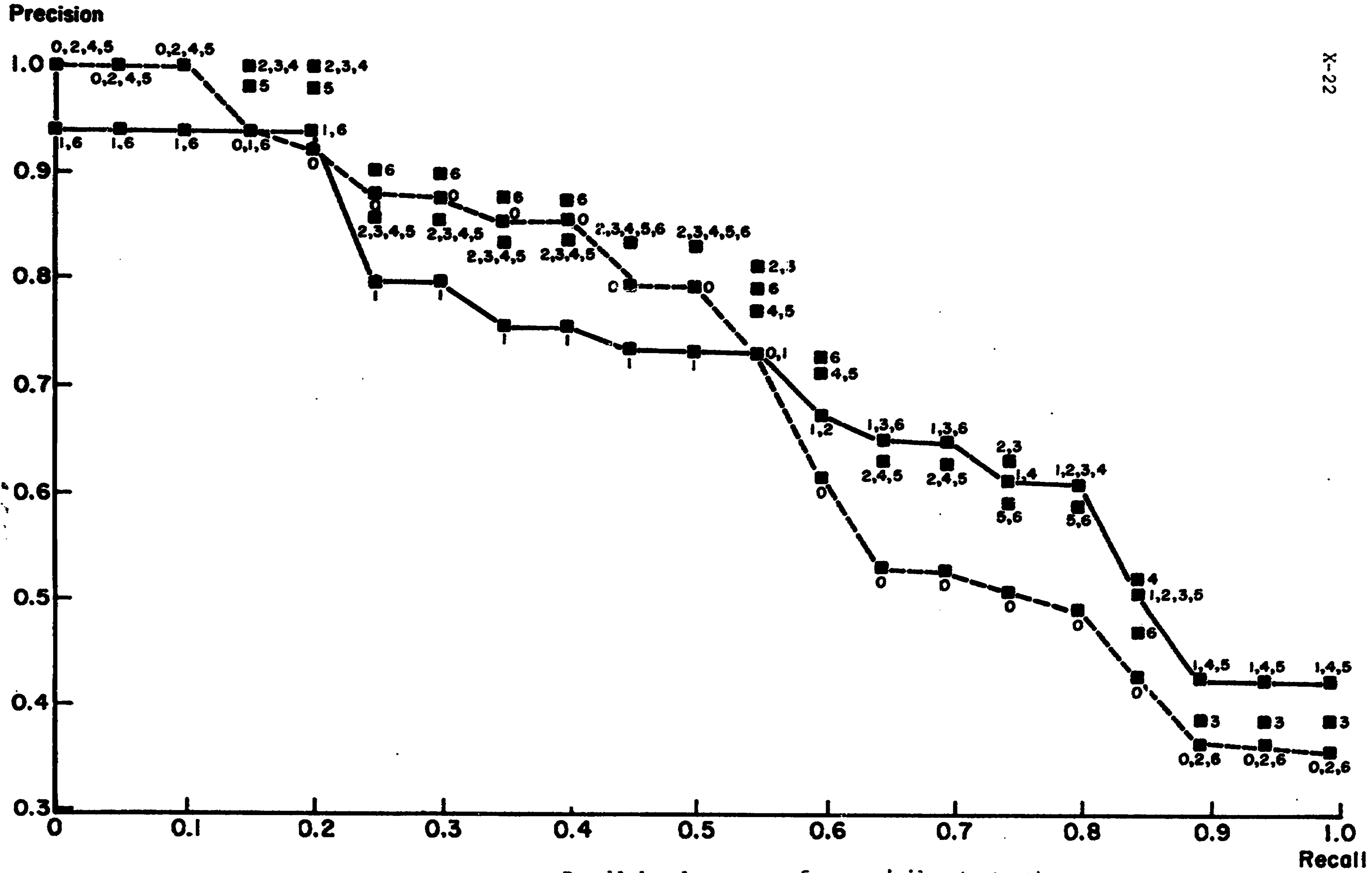
Recall level averages for Similar test set at oth iteration

Fig. 4



Recall level averages for similar test set
at 1st iteration

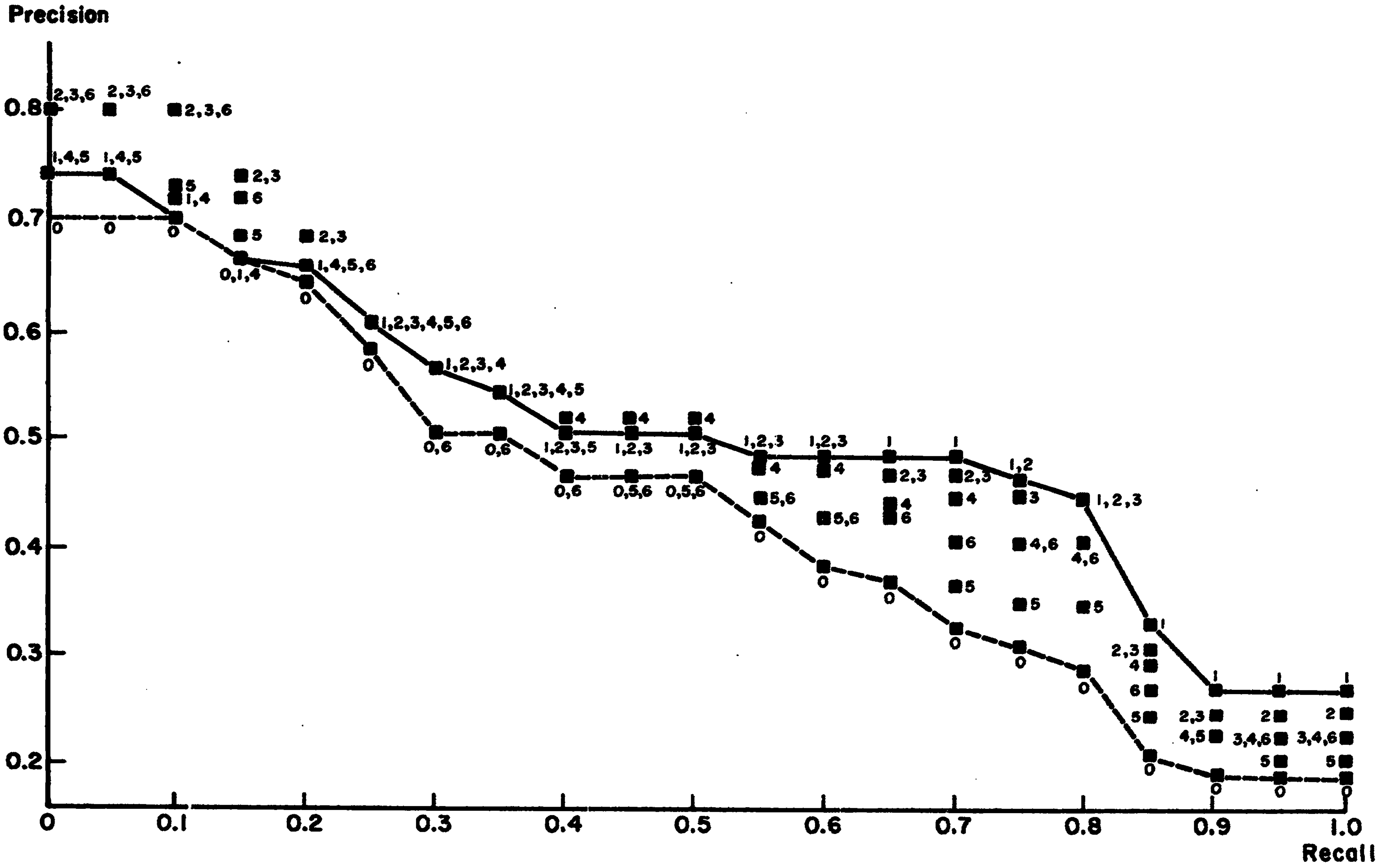
Fig. 5



Recall level averages for nonsimilar test set

at 1st iteration

Fig. 6



Recall level averages for nonsimilar test set at oth iteration

Fig. 7

which is superior to that of Set 0 but inferior to that of Set 1.

2) Nonsimilar queries in the oth iteration.
Slight cutoffs (Sets 2, 3, 4) are better than Set 1 at low recall levels (below 0.2) and equal to Set 1 at all other recall levels. Severe cutoffs seem to degrade the performance.

3) Similar queries in the first iteration.
Sets 1, 2, 3, 4 and 5 are practically equal in performance. The significance tests show that none is substantially better than the others. A very large cutoff like the one used for Set 6 produces a slight deterioration in performance. Compared with the standard search run (Set 0), all the six sets are by far superior.

4) Nonsimilar queries in the first iteration.
Sets of 2, 3, 4, 5 and 6 have essentially the same performance. They are all better than Set 1 at recall levels below 0.6.

From the above observation, one can reach the following conclusions: Except for the similar test set in the oth iteration, the cutoff of poor terms seems to maintain the performance of the standard DDSM. In the first iteration, the nonsimilar queries show a somewhat better performance. But too large a cutoff like Set 6 will tend to degrade performance. As a compromise, an intermediate cutoff like Set 3 is suggested. Considerable storage can be saved while effectiveness is still maintained.

The inferior performance of Sets 2, 3, 4, 5 and 6 in the oth iteration for similar queries is understandable. For most queries q_s, in the similar test set, (except queries 6, 9, 63, 78) one can find one or more queries, q_m, in the modification set with the following properties:

(i) q_s and q_m have almost the same relevant document set (say, D_1, D_2, ..., D_r)

(ii) Some negative discrimination value terms (say, C_1, C_2, ..., C_n) are in both q_s and q_m but not in the relevant documents (D_1, D_2 ..., D_r). These terms are very high frequency terms.

Since no cutoff occurs for Brauen's original PDSM, when q_m retrieves some of (D_1, ..., D_r) as relevant documents (say, D_1, ..., D_k) and modifies them, (C_1, ..., C_n) are added to these documents. Afterwards, when q_s in the simil.r test set is submitted, documents D_1, ..., D_k will have high correlation with q_s because C_1, C_2, ..., C_n appear in both q_s and D_1, D_2, ..., D_k. If a cutoff is used, C_1, C_2, ..., C_n are not added to D_1, D_2, ..., D_k when they are modified by q_m. This explains why Brauen's stand DDSM works especially well for similar queries in the oth iteraticn.

As mentioned above, terms C_1, C_2, ..., C_n are very high frequency negative discrimination value terms. Some of them even appear in half of the documents in the CRN4S DOCS collection, so they will probably appear in the queries very frequently. Actually Jones [5] showed that, for three independent collections, half of the query terms are high frequency terms. (Table 7) If many queries are submitted — in this experiment each document is modified only 1.77 times on the average — then each document will be modified many times, and these high frequency terms will enter into all document vectors in the long run. In case this happens, the effect of these terms will be negligible. Set 1 will then have roughly the same performance as set 2 through 5 for similar queries in the oth iteration.

Collections	Cranfield	INSPEC	Keen
No. of documents	200	541	797
No. of requests	42	97	63
No. of terms	712	1341	939
No. of frequent terms	96	73	50
Average No. of terms per request	6.9	5.6	5.3
Average No. of frequent terms per request	3.6	2	1.8

Term distribution statistics for three independent collections. (The last two rows show the ratios of frequent query terms)

Table 7

5. Conclusion

Sets 3 and 4 use NTERM = 2465. From Fig. 2 and Fig. 3, one can see that these sets actually achieve the same effectiveness as set 1. So, at least 4439 - 2465 = 1974 low discrimination value terms do not contribute much to retrieval effectiveness. For this value of NTERM. $\frac{(2969-2287)}{2969} \times 100\% = 23\%$ of the new terms are eliminated. A reasonable portion of the memory storage can therefore be saved.

This study also shows that the term discrimination value concept is acceptable. Slight cutoffs of nondiscriminators do not deteriorate retrieval performance. But neither do they improve effectiveness. One might suspect that additional work might be done in the theory of term discrimination values. Other approaches of defining term goodness on a sound theoretical ground is probably the most urgent. It is believed that, with a good judgment of the usefulness of terms, the proposed version of DDSM should lead to better results — both in performance and storage considerations.

References

[1] Brauen, T.L., Document Vector Processing, ISR-17, Report to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

[2] Bonwit, K. and Aste-Tonsmann, J., Negative Dictionary, Report ISR-18 to the National Science Foundation, Section VI, Department of Computer Science, Cornell University, October 1970.

[3] Crawford, R., Automatic Dictionary and Thesaurus Construction, Scientific Report No. ISR-22, Cornell University, to be published.

[4] Salton, G., The SMART Retrieval System, Prentice Hall, 1971.

[5] Jones, K.S., A Statistical Interpretation of Term Specificity and its Application in Retrieval, Journal of Documentation, Vol. 28, No. 1, March 1972.

The Use of Document Values for Dynamic Query Document Processing

A. Wong and A. van der Meulen

Abstract

In the field of document retrieval it might be advantageous to take into account the utility of documents in the collection; that is, the average usefulness of a specific document for a given user population in terms of satisfaction of the users' information need.

In this report the following two questions are investigated:

a) how to improve system performance by assigning so-called utility values to each document, and

b) how to base a document retirement policy on those quantities.

A feedback environment provides the possibility for automatically creating a list of utility values. These values are then based on the retrieval history of a document. That is, for all the queries for which a particular document is retrieved, user judgments about its relevancy are utilized to compute a quantity which reflects the usefulness of a document for the collection and its users.

Utility values may then be used during the retrieval process to promote the retrieval of satisfactory documents and suppress the retrieval of obsolete and mediocre documents. Another application of the availability of document quality values is a document retirement policy based on the utility value score in the collection. Documents

with low utility values may be placed in an auxiliary file to keep the main file more current and up to date.

1. Introduction

The performance of a document retrieval system is generally evaluated by means of two well known system parameters; recall and precision, which are based on the average user judgments about relevancy or non-relevancy of the retrieved documents.

Keeping track of user decisions might be quite beneficial for future users since provision can be made for detecting the quality of the judged documents. By automatically creating a new document property called the utility value, which represents the average judgment of the user population about a given document, it seems plausible that system performance can be improved by using this value in the retrieval process. Out of date documents or questionable publications, even if indexed in a proper way, will no longer be retrieved since their corresponding low utility values do prevent this.

The goal of this investigation is twofold:

a) system improvement by suppressing documents which are likely to be non-relevant and promoting relevant items in the course of the retrieval process;

b) system retirement by transferring out cf date and mediocre documents to an auxiliary file.

2. The Methodology

If the retrieval history of a set of queries in a system is known in the form of relevancy decisions of the user for the retrieved documents, one may use this information in deciding which documents are generally useful and which ones are not. In an on line system the relevancy decisions are rendered as part of the feedback procedures and the bookkeeping can be done in a very convenient way.

The procedure proposed is to assign to each document in the collection a so-called utility value which is set equal to 1 initially. This value is increased if a document is retrieved and found to be relevant, and decreased if retrieved and judged to be nonrelevant. The increment-decrement function is so chosen that the utility values range between 0 and 2 and are not able to exceed these values. (For a comprehensive description of this function, see [1], and Appendix 1.)

The utility value is then a system parameter which can be applied in the retrieval algorithm immediately since it reflects the utility as judged by the users. The new retrieval function Rf will be:

$$Rf = \cos(q,d) * U.V.$$

That is the product of utility value (U.V.) and the cosine correlation function (cos(q,d)).

3. The Organization of the Experiments

A. The Collection

The SMART retrieval system provides the Cranfield 1400 collection with 225 queries. This is a suitable collection for the experiments in that a maximum in updating of the utility values can be achieved. The query collection is split into an "updating collection" and a "test collection". The updating query collection is used to obtain utility values for all the retrieved documents while the test collection serves as a means for evaluating the influence of the retrieval algorithm mentioned in 2.

A set of 17 randomly chosen queries will serve as test collection. A comparison will be made between the retrieval results of the 17 queries without the use of and with the usage of utility values. In principal 208 queries were available to serve as updating collection. For practical reasons, however, a subset of 157 queries was actually used.

In an on-line retrieval system, which will be simulated in these experiments, the number of feedback iterations is likely to be at least one. That means that relevancy decisions for the first iteration are available for 157 queries. These retrieved documents will be used to determine utility values for the retrieved documents. For every query 30 documents are retrieved which requires in practice 30 relevancy decisions for each user. The number is chosen rather high in order to obtain a fair amount of updating and therefore probably more significant results.

B. The Updating Strategies

The most realistic way to implement an updating procedure is to do so dynamically. That is, each utility value changed after a search will be used in the retrieval algorithm and influences the retrievability of that specific document in later searches.

In the SMART environment the updating collection is run as a batch, the utility values being assigned afterwards and applied for the first time while running the test collection. (For a more complete discussion about static and dynamic updating of values see [1]). Similar experiments [2] concerning the updating of weighted dictionaries have shown that dynamic updating is superior to static results. If, for the experiments to be described in this report, the static approach turns out to be satisfactory a dynamic test might be useful at a later time.

Three different updating strategies are applied, namely

I The Straight Updating

For each query in the updating collection, the 30 retrieved documents are updated in accordance with the unmodified increment-decrement function mentioned. The utility values are increased if a retrieved document is relevant, and decreased otherwise. An arbitrary factor in the updating function which governs the stepsize of the increment or decrement is chosen equal to 8 (according to Sage, Ref. [3]). It is not a priori clear whether this value is optimal.

II The Balanced Updating

Since the average number of relevant retrieved documents for a query is approximately 5, an average of 25 documents will be decreased in utility value in strategy I. Thus for the whole collection the average utility value will become less than 1. Documents which are not updated have therefore a higher probability of being retrieved than updated ones since their utility value is still intact.

To eliminate this effect the number of decreased documents is kept equal to the number of increased ones. Thus in strategy II, all relevant documents included in the 30 retrieved ones are increased, and an equal number of the highest ranked nonrelevant documents is decreased.

III The Extended Balanced Updating

A disadvantage of the balanced updating strategy II is that the number of updatings is much smaller than in the case of method I. In strategy I, 30 utility values per query are updated, whereas for strategy II this number is reduced to about 10 (5 values increased and 5 decreased).

In order to combine the benefits of the maximum number of updatings (30) per query while preventing a decrease in the average utility value, Method III, the extended balanced updating is developed. In addition to the usage of all retrieved documents this method provides also an exact balancing. The sum of all the decrement steps is made equal for each query to the sum of all the increment steps. The decrement steps in particular will be chosen to be rank-dependent, that is, higher ranked nonrelevant documents are subject to a greater decrease in utility value than lower ranked ones. The rationale behind this is of course that high ranking nonrelevant documents should be suppressed. An extensive treatment of Method III is given in Appendix 2.

After applying these three strategies to the retrieval results of the query updating collection, one obtains a list of utility values which will be used in the retrieval algorithm for the query test collection:

$$Rf = \cos(q,d) \times U.V.$$

4. The Results

A. The Straight Updating

The results obtained using Method I (Fig. 1) are not promising. A possible explanation is the fact that almost all utility values are decreased. The drop in average utility value promotes the retrieval of documents that were not updated and degrades system performance. The retrieval algorithm consists of the product of utility value and cosine correlation. For this reason documents with utility value equal to 1 are ranked high, even when their actual correlation might be relatively low.

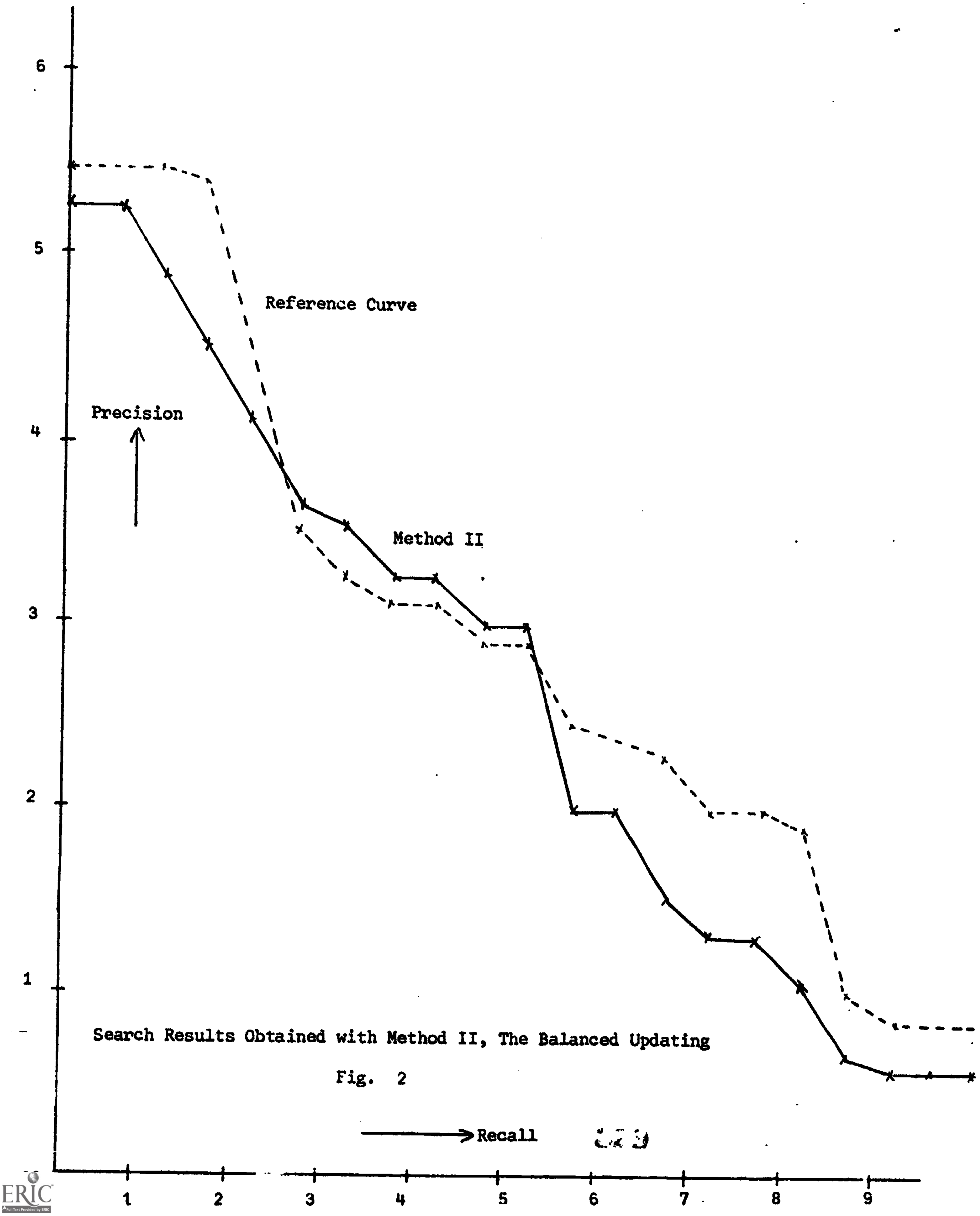
B. The Balanced Updating

The results obtained with Method II are considerably better (Fig. 2) than those of the unbalanced updating method. They show that the idea of balancing to keep the average utility value approximately equal to 1, is justified. Still a detoriation of system performance can be seen.

C. The Extended Balanced Updating

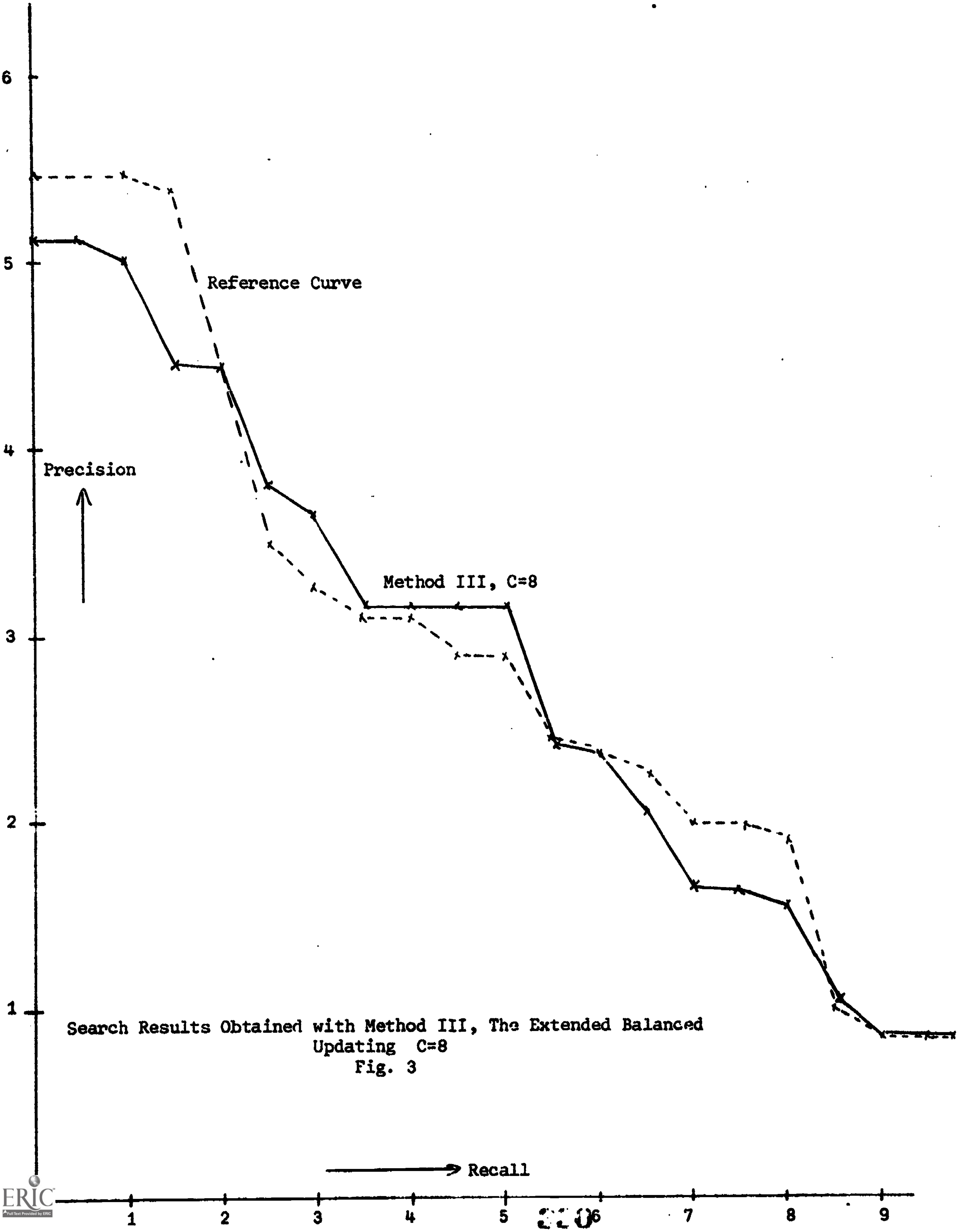
The results with this rather complicated algorithm (see Appendix 2) are better than those for both Method I and II (Fig. 3). They indicate that taking into account all the 30 available relevancy decisions per query is advantageous. In Method II the number of documents to be decreased in utility value is chosen equal to the number of documents to be increased. The stepsizes however are subject to the current size of the utility value, and therefore an exact balancing is not obtained. In Method III, however, the algorithm is devised such that for every query the sum of the increment steps is equal to the sum of the decrement steps, where all 30 retrieved documents are considered.

Since the results with Method III are the best obtained, and since no self-evident new philosophy for a fourth method could be found, a more complete analysis of the Sage increment-decrement



Search Results Obtained with Method II, The Balanced Updating

Fig. 2



Search Results Obtained with Method III, The Extended Balanced Updating C=8
Fig. 3

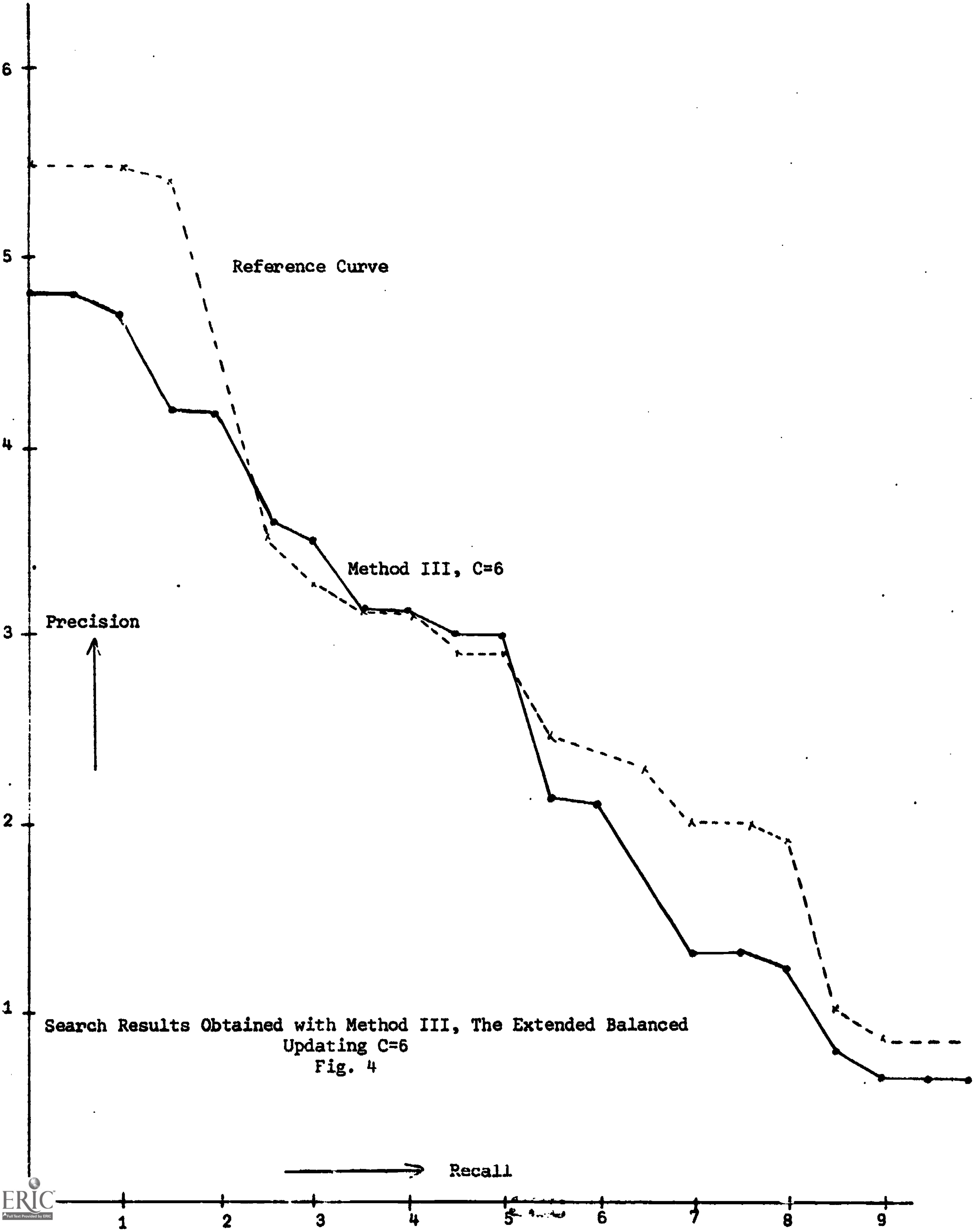
function was carried out. This function forms the basis of the iterative algorithm used in III, and as already mentioned, the constant C (see Appendix 1) which influences the stepsize has been set equal to 8 in all experiments. Decreasing this constant gives worse results, but increasing the value uields a remarkable system performance improvement (Fig. 4, 5, 6, and 7). A constant of approximately 17 turns out to be optimal in that the recall-precision curve for search r.sults (Fig. 6) has been lifted maximally compared with the reference curve.

It is a property of the algorithm that it is not particulary critical to changes in C. The starting value 8 was apparently much too small, that is, the updating step was too large (C occurs in the denominator). In the range 13 to 25 however, the retrieval results are not considerably affected by the change in C.

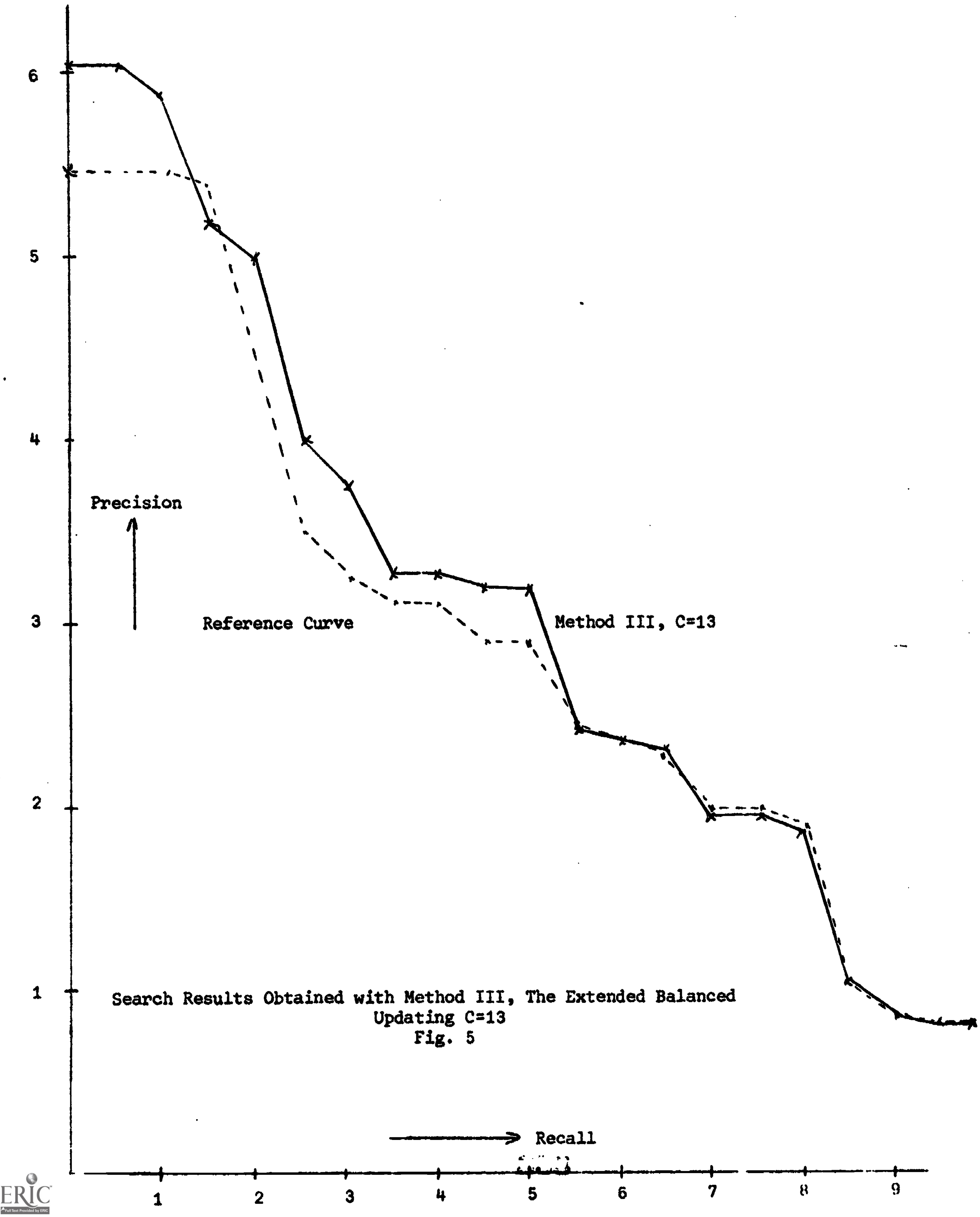
A Student T test was carried out in order to verify the signJficance of the results obtained with Method III, C=17. The outcome of the test (T statistic = 3.99 with 20 degrees of freedom) indicates that there is zero likelihood for the two sets of performance figures (reference versus Method III, C=17) to have originated in the same distribution.

5. Conclusion

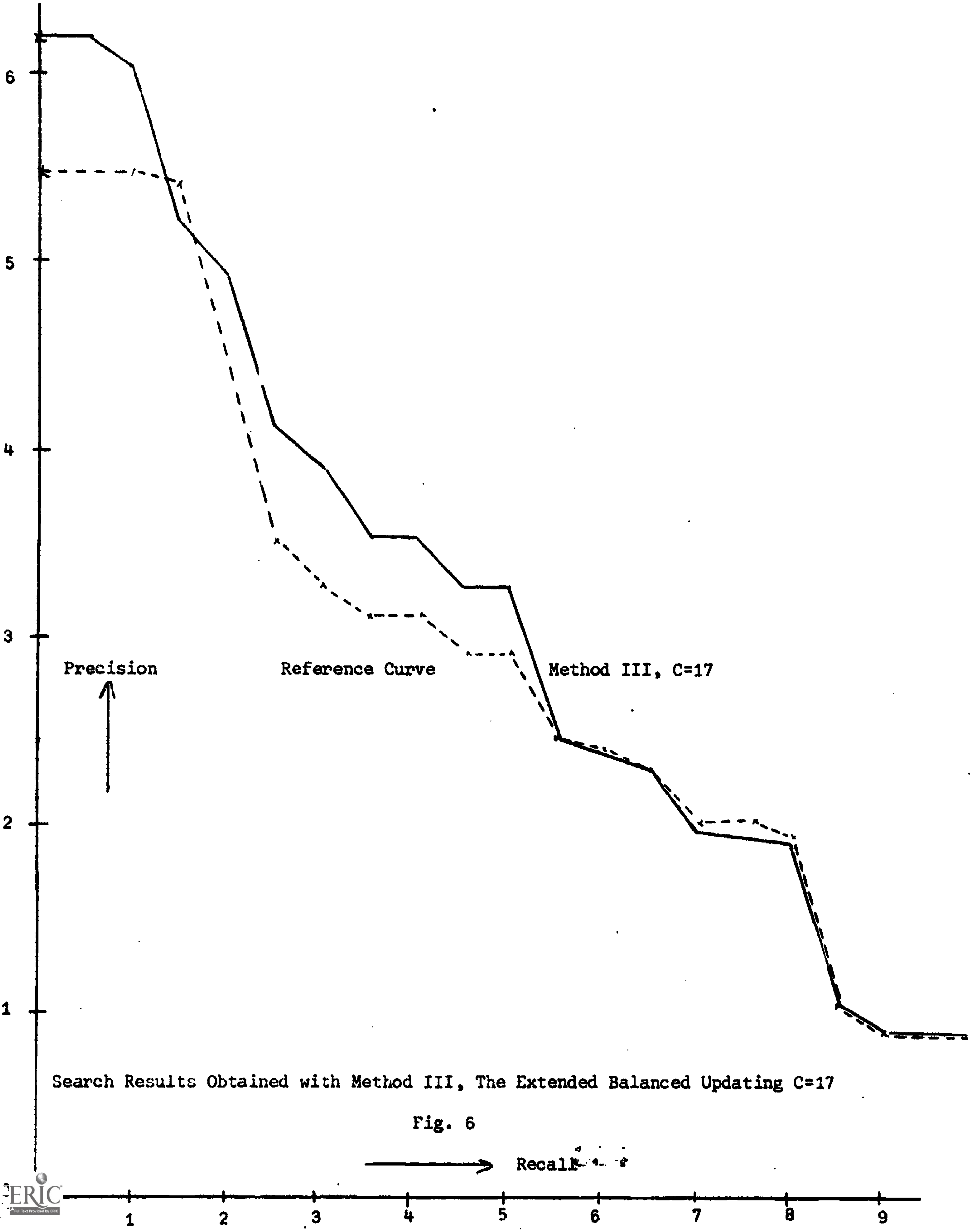
The experiments described in this report support unmistakably the usefulness of a new system parameter called utility value. The assignment of such a function to each document in the collection provides means for improving system performance. It has been shown that a careful approach of the concept is required, since it is found that



Search Results Obtained with Method III, The Extended Balanced Updating C=6
Fig. 4

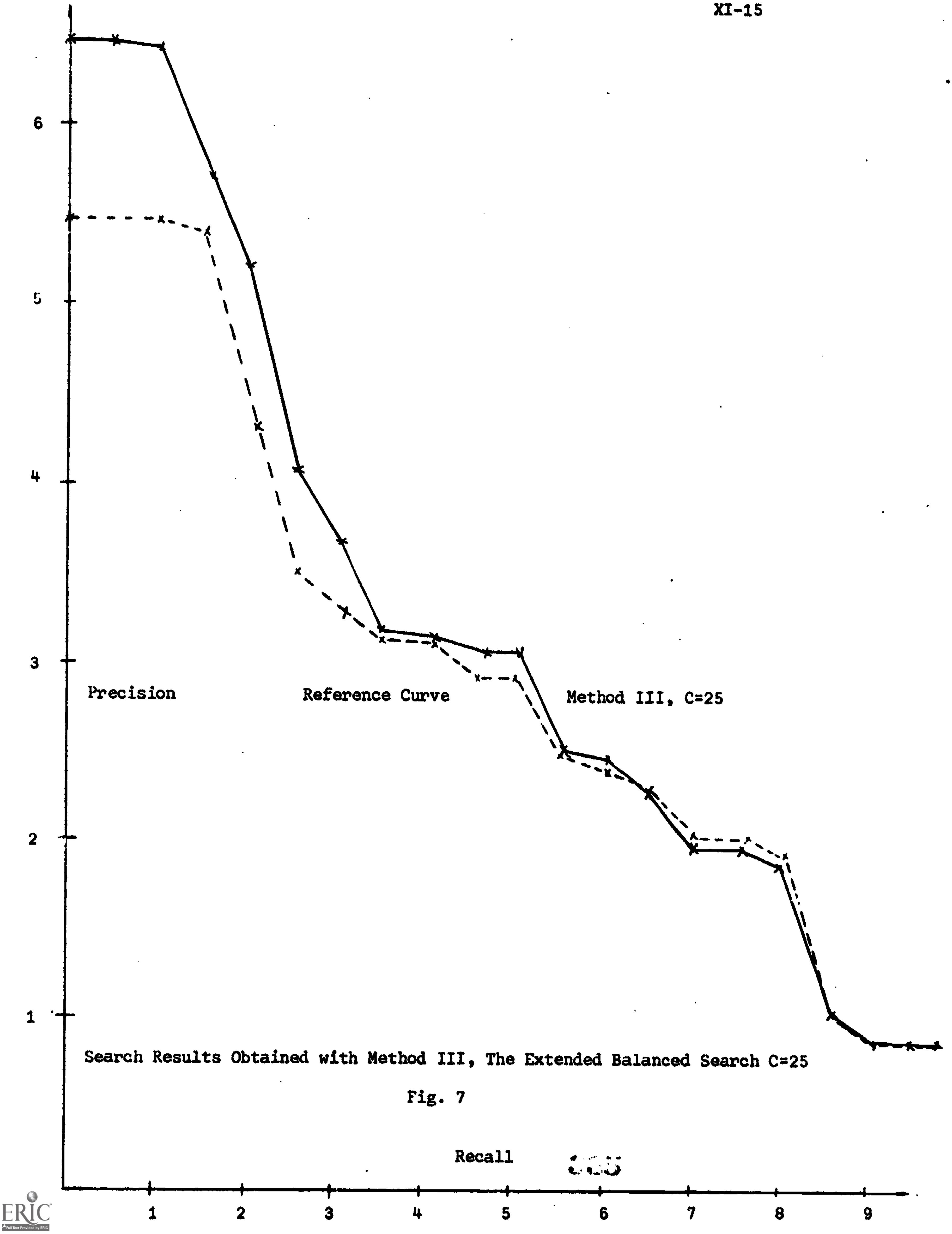


Search Results Obtained with Method III, The Extended Balanced Updating C=13
Fig. 5



Search Results Obtained with Method III, The Extended Balanced Updating C=17

Fig. 6



Search Results Obtained with Method III, The Extended Balanced Search C=25

Fig. 7

balancing — keeping the average utility value 1 — as well as the usage of heuristically determined constants are critical factors in applying a successful updating algorithm.

It is clear that any document retirement policy based on document values can only be justified is such values have a realistic meaning, that is, after applying them system performance has to be improved. This investigation has proved that the latter may be possible.

References

1 A. Wong, R. Peck, and A. van der Meulen, An Adaptive Dictionary in a Feedback Environment, Term Project Report, Spring 1972, Department of Computer Science, Cornell University.

2 A. van der Meulen, Dynamic versus Static Information Values, to be published.

3 C.R. Sage, R.R. Anderson, and D.P. Fitzwater, Adaptive Information Dissemination, American Documentation, Vol. 16, No. 3, July 1965.

Appendix 1: The Increment-decrement Function

Initially, the utility value of each document is set equal to one. From that point the value is increased (or decreased) according to the relevancy decisions of the documents retrieved in response to each query.

The specific increment-decrement function chosen is the one proposed by Sage [3], herein referred to as the "sine-function" because of its resemblance to the regular sine. If i is the retrieved document, define:

v_i = the utility value of document i
(initially set equal to one)

v_i^* = the utility value of document i after updating

x_i = arc sin $(v_i - 1)$, the transposed utility value.

Then, $v_i = 1 + \sin(x_i)$

and similarly v_i^* is calculated by

$$v_i^* = 1 + \sin(x_i \pm \Delta x_i) \tag{1}$$

where Δx is a function of the old utility value, calculated as follows:

$$\Delta x_i = \frac{\pi/2 - |x_i|}{C} \tag{2}$$

C is an arbitrary constant.

Δx_i is added in equation (1) if the retrieved document i is judged relevant by the user; or it is subtracted in the utility value calculation if the corresponding document is judged non-relevant.

Appendix 2: The Extended Balanced Updating

Let q be a query which retrieves 30 documents and let δ_i be the increment of decrement stepsize of document i according to the Sine function. According to equation (1) of Appendix 1, δ_i is given by

$$\delta_i = | v_i^* - v_i | = |\sin x_i - \sin(x_i \pm \Delta x_i) |$$

The sum of all increment steps for query q can be given as

$$S_+ = \sum_{i \varepsilon \text{Rel doc}} \delta_i$$

Since the number of non-relevant documents retrieved by query is considerably larger (25 to 5) than the number of relevant ones, the decrement steps of the utility values for the nonrelevant documents should be made smaller. A rank dependent decrement step is chosen such that low ranked nonrelevant documents will be decreased only slightly. Moreover, the function is adjusted by a free parameter such that the sum of all decrement steps is going to be equal to $S_+ \cdot S_-$ is given by

$$S_- = \sum_{j \varepsilon \text{non-rel doc}} \Delta_j$$

where Δ_j the actual decrement of the nonrelevant document j, is chosen to be a linear function f_j of the retrieval ranks, according to the equation

$$\Delta_j = \frac{\delta_j}{f_j} = \frac{\delta_j * 29}{D(rj - 1) + 30 - rj} \quad j \varepsilon \text{non-rel doc} \qquad (1)$$

D is constant for one query but its value is changed for the next one because the number and ranks of relevant and nonrelevant documents change from query to query. The implementation of this method requires an iterative algorithm in order to find D for each query such that

$$S_- = S_+ \qquad (2a)$$

or

$$\sum_{j\varepsilon\text{Non-rel Doc}} \Delta_j = \sum_{i\varepsilon\text{Rel Doc}} \delta_i \qquad (2b)$$

Consider first an illustration of equation (1) in order to render the method clearer. A nonrelevant document retrieved in rank 1 will be decreased by δ_j, that is, the unmodified Sine function (f_j=1). A nonrelevant document retreived in rank 30 will be decreased by $\frac{\delta_j}{D}$ (f_j = D). All nonrelevant documents with ranks in between will be decreased subject to the linear function (1).

The value of D on the other hand is determined by equation (2) and cannot be explicitly computed. An initial value for D must be chosen and equation (2) can be evaluated. D has to be changed in an iterative way until $|S_- - S_+| \leq \varepsilon$, where ε is small.

The procedure guarantees an almost exact balancing of the amount of increase and decrease of utility values per query.

Following is an example illustrating how the document values are changed due to one query.

30 documents are retrieved by a query. Some are found relevant. The document values of these 30 documents, after a number of previous updatings, will be changed according to the relevancy decision of the present query (see Table 1).

S_+ the sum of the increased document values for the relevant documents 320, 467, 322, and 321, is found to be 0.3760, using equation

Doc. No.	Rank	Old Doc. Value	New Doc. Value
320 R	1	1.0941	1.1818
476 R	3	0.9980	1.0920
322 R	4	0.9879	1.0820
321 R	7	0.9959	1.0898
107	2	0.9980	0.9325
478	5	1.0000	0.9645
479	6	0.9959	0.9653
734	7	0.9980	0.9740
190	9	0.9889	0.9672
452	10	0.9962	0.9763
1251	11	0.9697	0.9518
1163	12	0.9939	0.9770
1149	13	0.9817	0.9661
422	14	0.9979	0.9832
255	15	1.0916	1.0886
47	16	1.0564	1.0438
1162	17	0.9970	0.9847
837	18	0.9780	0.9665
1209	19	1.0000	0.9889
150	20	0.9883	0.9778
34	21	1.0000	0.9899
1254	22	1.0000	0.9903
979	23	0.9914	0.9821
626	24	0.9981	0.9892
1235	25	1.0000	0.9914
1370	26	0.9360	0.9281
538	27	0.9959	0.9879
425	28	0.9814	0.9738
818	29	1.0767	1.0686
363	30	1.0000	0.9928

Alterations in Document Values

Table 1

D	5	7	9	11	13
S	0.8614	0.6735	0.5573	0.4775	0.4190
$\|S_+ - S_-\|$	0.4854	0.2975	0.2813	0.1015	0.0430

The Variation of S with respect to D

Table 2

1 of Appendix 1.

Δ_j's are found interatively by changing the value of D, such that S_+ is equal to S_- to within a tolerance, which is 0.05 in this case.

For the variation of S_- for different values of D, see Table 2.

For the present query, D = 13 is used.

Automatic Document Retirement Algorithms

K. Sardana

Abstract

Some existing and proposed algorithms for automatic document retirement in a retrieval system are analyzed for their computational complexities and their effects on storage costs and the retrieval performance of the system. It is found that various retirement algorithms exhibit almost equivalent performance, especially at high recall; therefore, it is preferable to use those algorithms which provide low cost bounds.

1. Introduction

Two automatic document retirement algorithms have been proposed by Tai and Yang (referred to as TY in the sequel) [1] and Beall and Schnitzer (referred to as BS in the sequel) [2]. It is, however, not clear what algorithm should be used in practice, what costs are involved in executing the algorithms, what savings are obtained and at how much loss or gain in retrieval performance, etc. The idea of the present study is to look into these questions with special reference to TY algorithm. Some more algorithms are proposed and overall comparisons are made between different algorithms when used with and without document vector modification (DVM). [4]

2. The Algorithms

Both the TY and BS algorithms for document retirement are based on utilizing users' relevancy judgments in updating the documents. The algorithms that we propose also rely on users' relevancy judgments. We assume that relevancy judgments on retrieved documents for various queries are available as input to the retirement algorithms. The computational complexity of the algorithms will consist of the costs of modifying the documents, computing "use indices" for documents, etc. for the express usage of the algorithms and the cost of making retirement decisions.

The general philosophy used here is to give the algorithms and their computational complexities. As we will not be able to give proofs of correctness of these heuristic algorithms, we resort to experimental methods in the next section to evaluate the performance of the methods in an experimental environment. Finally overall comparisons between algorithms are mad .

Criteria of Evaluation of Algorithms

In determining the computational complexity of the algorithms, we assume the following:

a) Asymptotic complexity will be used so that a machine independent cost analysis can be done. However, constants will be considered when finer decisions are involved.

b) The model of the computing device is a random access machine which assumes that enough core memory is available for the entire program to fit in.

c) The worst case computational complexity will be derived rather than expected complexity as the former is easier to get hands on.

d) The basic steps in the computation to be considered for time complexity are additions (adds), multiplications (mults) and comparisons (comps).

e) The cost criterion is uniform rather than logarithmic i.e. a unit of each kind of operation will have some uniform cost regardless of the size of the numbers involved. A detailed discussion on choosing the above criteria may be found in [3].

Input: A set of documents {D} and a set of queries {Q} and users' relevancy judgments on {D} for {Q}.

Output: A set of documents $\{D'\} \subseteq \{D\}$ to be retired to a secondary store such that future retrieval processing with remaining {D} - {D'} documents using future queries is expected to be overall efficient considering the space saved by retirement, gain or loss in system performance and cost of execution of the algorithm.

Some Desirable Features of Algorithms

To achieve the above mentioned goal of retiring documents, the following features (among others) of the algorithms seem to be desirable:

a) Irretrievable documents or documents not relevant to any query should be retired. This, however, assumes that the indexing is perfect.

b) The selection of various parameters for the algorithm should be pretty straightforward in any practical implementation.

c) The algorithm should not assume any unnecessary attributes of the document space, e.g. documents with same average weights, etc.

We note that each of the TY or BS algorithms does not have one or the other of the above features.

Methods

In the following algorithms, we mention the step to retrieve the documents and to do DVM for a query. However, the costs of these steps are not considered in determining the computational complexity of a retirement algorithm as these ster are not part of the retirement algorithm, per se.

It may be argued that the analysis of computational complexity of such isolated algorithms does not make much sense insofar as the environment in which they are going to be used is known. Thus, for instance, one may consider the effect of complexity of a retirement algorithm on the asymptotic complexity of the overall retrieval process. The cost of matching a query with n documents, each document having on the average m concepts, is O(mn) and the cost of selecting top r documents for showing to the user is O(n) using the median algorithm [5] implying the overall retrieval process to be O(mn)/query. Since most of the additional algorithms, like the retirement algorithms considered here, cost less than O(mn)/query (see later), the asymptotic complexity of the overall retrieval process does not change. It may, then, be concluded that the analysis of such algorithms is not important. The viewpoint taken here is that in document retrieval, where the cost of answering a query is quite high, the reduction of costs of all sub-algorithms should be considered important.

The algorithms are expressed in pseudo-Algol for clarity and ease in deriving the computational complexities. For convenience, we assume a kind of macro facility available in the language with four keywords: "<u>defmacro</u> X" defines a macro named X, the code between <u>begmacro</u> and endmacro is body of the the macro named X (assuming "defmacro X" precedes this). The body of the macro X is textually substituted for call "macro X".

A1) Algorithm TY (Tai and Yang)

The TY algorithm [1] shown in Fig. 1(a) is basically the following, suitably modified to work with a batch of queries (BATCHSIZE is the number of queries in a batch and BATCHNUM is the number of batches to be processed):

a) For each query q_{ij} in a batch, retrieve r number of documents. Form the sets REL_{ij}, $NREL_{ij}$ and BOT_{ij} consisting of top RELTOP relevant retrieved, at most NONLIM nonrelevant among top NONTOP retrieved and bottom-most ranking documents NUMBOT in number respectively.

b) After doing DVM (optional) using the queries in a batch. multiply the weights of concepts of document vectors in REL_{ij} $NREL_{ij}$ and BOT_{ij} by constants FR(>1), FN(<1) and FB(<1) respectively.

c) After every document vector multiplication by FN or FB, compute the average weight AVGWT of the document and retire it if AVGWT ≤ CUTOFF, some chosen constant.

The idea of this algorithm is to reward the top relevant retrieved documents and to penalize the top nonrelevant retrieved and the bottom-most ranking documents for each query by respectively increasing or decreasing all the weights of the concepts of the documents by the same factor at a time. The information on the usefulness of a document is thus carried in the weights of the vector itself. When the average weight per concept of a document falls below a chosen threshold, meaning that the document has been penalized more than it has been rewarded, the document is retired.

Computational Complexity:

Let

n = total number of documents (roughly 400-1500) in the special subject area of the query; thus for a query in Astrophysics, n is the number of documents in this particular subcollection rather than the whole Physics library.

q = total number of queries in the same area = $O(n)$ say.

m = average number of concepts/document.

r = number of documents retrieved.

We also assume that

i) $|REL_{ij}| + |NREL_{ij}| + |BOT_{ij}| \simeq r$, where $|X|$ denotes the cardinality of set X. This is not an unreasonable assumption to make and conforms in practice.

ii) $|REL_{ij}| \simeq cr$ for some constant $c(\leq 1)$.

Let us first determine the cost associated with SETUP code of Fig. 1(b), used in line (7) of algorithm TY of Fig. 1(a). Costs associated with lines (8), (9) and (10) are constant. Cost of line (6) is ~r comps/query (~ stands for approximately) because from the given ranked list of documents obtained from line (4), this many comparisons may be needed to form sets REL_{ij} and $NREL_{ij}$.

Next we determine the cost of T&Y code of Fig. 1(c) used in line (10) of Fig. 1(a). Lines (4), (7) and (13) together cost ~m mults/documents/query i.e. ~mr mults/query over all documents in sets REL, NREL and BOT. Similarly cost of lines (8), (9), (14) and (15) over documents in sets NREL and BOT is ~mr adds, ~r mults (actually divisions to calculate AVGWT) and ~r comps per query.

```
Line #   Cost

 1                            procedure RETIRE comment Tai and Yang Algorithm
 2                              begin
 3                                for j ← 1 step 1 until BATCHNUM do
 4                                  begin
 5                                    REL = {φ}; NREL = {φ}; BOT = {φ};
 6                                    for i ← 1 step 1 until BATCHSIZE do
 7  ~r comps/query                    macro (SETUP);
 8                                    comment optionally do DVM in next step using ($REL_{ij}$,$q_{ij}$)
                                              pairs, BATCHSIZE in number
 9  ~mr+r mults/query                 macro (DVM);
10  ~mr adds/query                    macro (T&Y);
    ~r comps/query
11                                  end
12                              end
```

Algorithm TY, Tai and Yang's Algorithm

Fig. 1(a)

```
Line #   Cost

 1                           defmacro SETUP
 2                           begmacro
 3                             begin
 4                               retrieve r documents for ith query in jth batch
 5                                        (call this query $q_{ij}$);
 6  ~r comps/query               form sets $REL_{ij}$, $NREL_{ij}$ and $BOT_{ij}$;
 7                               comment · operation below denotes concatenation
 8                               REL = REL · $REL_{ij}$;
 9                               NREL = NREL · $NREL_{ij}$;
10                               BOT = BOT · $BOT_{ij}$;
11                               save ($REL_{ij}$, $q_{ij}$) pair on a list;
12                             end
13                           endmacro
```

Definition of Macro SETUP

Fig. 1(b)

```
Line #   Cost

  1                          defmacro T&Y
  2                            begmacro
  3                              for each document D in REL do
  4                                multiply all the weights of the concepts by a constant fr;
  5                              for each document D in NREL do
  6  ~mr mults/query               begin
  7   (steps 4,7,13)                 multiply all the weights of the concepts by a constant fn;
     ~mr adds/query
  8  +~r mults/query                 calculate avg. wt., AVGWT of the concepts;
  9   (steps 8,14)                   if AVGWT ≤ CUTOFF then retire this document D;
     ~r comps/query
 10   (steps 9,15)                 end
 11                              for each document D in BOT do
 12                                begin
 13                                  multiply all the weights of the concepts by a constant fb;
 14                                  calculate avg.wt., AVGWT of the concepts;
 15                                  if AVGWT ≤ CUTOFF then retire this document D;
 16                                end
 17                            endmacro
```

Definition of Macro T&Y

Fig. 1(c)

```
Line #   Cost

  1                          defmacro DVM
  2                            begmacro
  3                              for k ← 1 step 1 until BATCHSIZE do
  4                                begin
  5                                  for each document D in REL_kj do
  6  ~mr comps/query                   for each concept C(l) belonging to D and q_kj do
     ~mr mults/query
  7  ~2mr adds/query                     W(l)' = W(l) + α*(BIG-W(l));
  8                                    comment W(l)' and W(l) are weights of concept
                                         C(l) before and after the operation.  α and
                                         BIG are constants defined by Brauen [4]
  9                                end
 10                            endmacro
```

Definition of Macro DVM

Fig. 1(d)

Summing up, the total cost of algorithm TY is

$$\begin{array}{llll} \sim r + \sim r & = \sim 2r & \text{comps/query} & = O(r) \\ \sim mr + \sim r & = \sim r + mr & \text{mults/query} & = O(mr) \\ \sim mr & = \sim mr & \text{adds/query} & = O(mr) \end{array} \qquad (1.1)$$

or O(mr) operations/query.

A2) Algorithm BS (Beall and Schnitzer)

The BS algorithm [2] shown in Fig. 2 works as follows:

a) For each query q_{ij} retrieve r documents. Using some chosen parameters, form the sets REL1_{ij}, REL2_{ij}, NREL_{ij} and BOTZERO_{ij} consisting of the top relevant retrieved, middle ranking relevant retrieved, top nonrelevant retrieved and the bottom-most ranking or zero correlating documents.

b) Using these sets, do a special kind of DVM by which concepts common to query q_{ij} and each document in REL1_{ij} or REL2_{ij} are increased and concepts common to query q_{ij} and each document in NREL_{ij} or BOTZERO_{ij} are decreased in weight at different rates.

c) After processing a number of queries in this fashion (even though this is not explicitly stated in [2]), if a document has (i) less than NUM concepts of weight greater than MIN1 and (ii) the average weight of the document is less than MIN2, then this document is retired. NUM, MIN1 and MIN2 are parameters chosen for the algorithm.

By this algorithm, the information about the usefulness of a document is carried in a more or less ad hoc manner, in the concepts common to the queries used for processing and the document. The retirement decision is made at the end using a careful examination of each document vector.

```
Line #   Cost
 1                          procedure RETIRE comment Beall and Schnitzer Algorithm
 2                            begin
 3                              for j ← 1 step 1 until BATCHNUM do
 4                                begin
 5                                  for i ← 1 step 1 until BATCHSIZE do
 6                                    begin
 7                                      retrieve r documents for ith query in jth batch
 8                                                 (call it query $q_{ij}$);
 9  ~r comps/query                      form sets $REL1_{ij}$, $REL2_{ij}$, $NREL_{ij}$, $BOTZERO_{ij}$;
10                                      save quintuple ($REL1_{ij}$, $REL2_{ij}$, $NREL_{ij}$, $BOTZERO_{ij}$, $q_{ij}$)
                                                   on a list;
11                                    end
12                                  comment do the DVM using quintuples saved, BATCHSIZE in number
13                                  for i ← 1 step 1 until BATCHSIZE do
14            ↑                       begin
15            |                         for each document D in $REL1_{ij}$ do
16            |                           if concept C(k) belongs to D and $q_{ij}$ then
17            |                             $W(k)' = W(k) + \alpha_1*(BIG-W(k))$;
    $\sim\frac{r}{2}\cdot m$ comps/query
18  ~3mr mults/query                    for each document D in $REL2_{ij}$ do
19  ~3mr adds/query                       if concept C(k) belongs to D and $q_{ij}$ then
20            |                             $W(k)' = W(k) + \alpha_2*(BIG - W(k))$;
21            |                         for each document D in $NREL_{ij}$ do
22            |                           if concept C(k) belongs to D and $q_{ij}$ then
23            ↓                             $W(k)' = W(k) - (W(k)/N + 1)$;
24                                      for each document D in $BOTZERO_{ij}$ do
25  ~mr comps/query                       if concept C(k) belongs to D and $q_{ij}$ then
26  ~mr mults/query                         $W(k)' = W(k) * \beta$;
27                                    end
28                                end
29                                  comment make retirement decisions, n=no. of documents,
                                                    m=avg. no. of concepts in document
30                              for i ← 1 step 1 until n do
31                                    begin
32                                      INUM ← 1; AVGWT ← 0
33                                      for j ← 1 step 1 until m do
34                                        begin
35  ~mn comps/q queries                     if weight of concept C(j) in document i, $W(C(j)_i)$
36  ~mn adds/q queries                          > MIN1 then INUM ← INUM + 1;
37  ~mn adds/q queries                      AVGWT ← AVGWT + $W(C(j)_i)$
38                                        end
39  ~n mults/q queries                    AVGWT ← AVGWT/m
40  ~2n comps/q queries                   if INUM < NUM and AVGWT < MIN2 then
41                                            retire the document i;
42                                    end
43                            end
```

Algorithm BS, Beall and Schnitzer's Document Retirement Algorithm

Fig. 2

In the BS algorithm, the special kind of DVM is an integral part of the algorithm; this is actually a drawback since the algorithm cannot be used unless DVM is also desired. The cost of this algorithm including DVM cost (from cost column in Fig. 2) is

$$\begin{aligned} &\sim r + \frac{r}{2} \cdot m + mr + \frac{mn}{q} + \frac{2n}{q} &&= \sim\frac{3}{2}mr + r + \frac{(m+2)n}{q} &&= O(mr) \text{ comps/query} \\ &\sim 3mr + mr + \frac{n}{q} &&= \sim 4mr + \frac{n}{q} &&= O(mr) \text{ mults/query} \quad (1.2) \\ &\sim 3mr + \frac{mn}{q} + \frac{mn}{q} &&= \sim 3mr + \frac{2mn}{q} &&= O(mr) \text{ adds/query} \end{aligned}$$

To make a fair comparison of this algorithm with the previous one, we include the cost of doing DVM along with TY algorithm also. Then the total cost of TY algorithm (from equations (1.1)) and DVM cost (from the cost column in Fig. 1(d)) is

$$\begin{aligned} &\sim 2r + mr &&= \sim mr + 2r &&= O(mr) \text{ comps/query} \\ &\sim mr + r + mr &&= \sim 2mr + r &&= O(mr) \text{ mults/query} \qquad (1.3) \\ &\sim mr + mr &&= \sim 2mr &&= O(mr) \text{ adds/query.} \end{aligned}$$

Thus the asymptotic time complexity of BS algorithm considering the DVM cost is O(mr), same as that of TY algorithm. But considering constants in equations (1.2) and (1.3), BS algorithm seems really the costlier of the two.

A3) Algorithm S1

The algorithm S1 (Figs. 3(a), 3(b)) is a variant of TY algorithm and is described below.

a) Initialize the value of the separately assigned storage location USENDX of each document to INIT.

```
Line #   Costs

 1                              procedure RETIRE comment variation of TY algorithm, S1 & S2
 2                                begin
 3                                  for i ← 1 step 1 until n do USENDX(i) = INIT;
 4                                    comment n = number of documents
 5                                  for j ← 1 step 1 until BATCHNUM do
 6                                    begin
 7                                      REL = {φ}; NREL = {φ}; BOT = {φ};
 8                                      for i ← 1 step 1 until BATCHSIZE do
 9  ~r comps/query                        macro (SETUP);
10                                      comment optionally do DVM in next step using
11                                              (REL_ij, q_ij) pairs, BATCHSIZE in number
12                                        macro (DVM);
13 Alg. S1:~r mults/query                if algorithm S1 is desired  then macro (S1)
   Alg. S2:~r adds/query                                             else macro (S2);
14                                    end
15                                  for i ← 1 step 1 until n do
16 2n/q comps/query                   if (USENDX(i) ≤ CUTOFF or USENDX(i) = INIT) then
17                                             retire the document i;
18                               end
```

Algorithms S1 and S2

depending upon the algorithm desired
corresponding named macro
is expanded in line 13 above

Fig. 3(a), 4(a)

```
Line #   Cost;

 1                            defmacro S1 comment for algorithm S1
 2                              begmacro
 3     ↑                          for each document i in REL do
 4     |                                    USENDX(i) ← USENDX(i) * FR;
 5     |                          for each document i in NREL do
 6 ~r mults/query                           USENDX(i) ← USENDX(i) * FN;
 7     |                          for each document i in BOT do
 8     ↓                                    USENDX(i) ← USENDX(i) * FB;
 9                             endmacro
```

Definition of Macro S1 for Use in Algorithms S1

Fig. 3(b)

b) Retrieve r documents for query q_{ij}. Form the sets REL_{ij}, $NREL_{ij}$ and BOT_{ij} just like in TY algorithm.

c) After doing the DVM (optional) using the queries in a batch, multiply the USENDX of a document by FR(>1), FN(<1) or FB(<1) according as this document appears in the set REL_{ij}, $NREL_{ij}$ or BOT_{ij} respectively.

d) After processing a BATCHNUM number of batches of queries using steps (b) and (c) repeatedly, if a document's USENDX = INIT or USENDX ≤ CUTOFF, some chosen parameter, then this document is retired. Go to step (a).

Note, here DVM does not directly take part into retirement decisions as in TY algorithm, but does so indirectly by choosing which documents get retrieved or not thus affecting their USENDX values. The main difference between TY algorithm and the present one is: The former uses AVGWT of the documents to denote the useful index for a document and, therefore, since the document vectors are modified by DVM and the multipliers FR, FN or FB, AVGWT needs to be computed for each document vector before making a retirement decision. The latter algorithm uses a location USENDX for each document and its value gets modified by multipliers FR, FN or FB while not by DVM directly. The retirement decision is based on USENDX values.

Another difference is that as shown, the TY algorithm makes retirement decisions after processing every batch of queries (5 used here) while S1 algorithm does so after processing a set of queries (125 used here). But both algorithms may be adjusted to any retirement frequency, in which case the asymptotic cost of TY algorithm does not change while the cost of S1 algorithm may approach O(n) when retirement decision is made after processing every query.

Computational Complexity:

As shown in the cost column of Fig. 3(a), the cost of this algorithm is

$$\begin{aligned} &\sim r + \sim\frac{2n}{q} \quad \text{comps/query} = 0(r) \text{ comps/query} \\ &\sim r \quad \text{mults/query} = 0(r) \text{ mults/query} \end{aligned} \tag{1.4}$$

or 0(r) operations/query since q = 0(n).

This algorithm is thus asymptotically better than TY algorithm in time complexity by a factor of m, the average number of concepts in a document vector. However the space complexity has been increased by 0(n) in using probably one computer word or halfword USENDX location for each of the n documents. This is reasonable considering that the space required is on a cheap off-line device while the time saved decreases the important on-line response time.

A4) Algorithm S2

This algorithm (Figs. 4(a), 4(b)) another variant of TY algorithm, resembles algorithm S1 closely. Here in step (b) the USENDX of a document is increased or decreased by constant values by additions or subtractions rather than by multiplications (as is done in algorithm S1) whenever a document ends up in the set REL_{ij} or $NREL_{ij}$ or BOT_{ij}.

Computational Complexity:

From the cost column of Fig. 4(a), the time complexity of this algorithm is

$$\begin{aligned} &\sim r + \sim\frac{2n}{q} \quad \text{comps/query} = 0(r) \text{ comps/query} \\ &\sim r \quad \text{adds/query} = 0(r) \text{ adds/query} \end{aligned} \tag{1.5}$$

```
Line #     Cost
13.1                      defmacro S2 comment this macro is for algorithm S2
13.2        ↑               begmacro
13.3        |                 for each document i in REL do
13.4        |                   USENDX(i) ← USENDX(i) + FREL;
13.5        |                 for each document i in NREL do
13.6   ~r adds/query            USENDX(i) ← USENDX(i) + FNREL;
13.7        |                 for each document i in BOT do
13.8        ↓                   USENDX(i) ← USENDX(i) + FBOT;
13.9                        endmacro
```

Definition of Macro S2 for Use in Algorithm S2

Fig. 4(b)

or O(r) operations/query.

Remarks similar to algorithm S1 apply here also.

A5) Algorithm S3

Algorithm S3, Fig. 5, differs from algorithm S2 in step (b) in two respects:

i) Instead of assigning a fixed positive USENDX to a document appearing in set REL_{ij} algorithm S3 assigns a variable index $\frac{1}{p}$ depending upon the number p of relevant retrieved documents (i.e. $p = |REL_{ij}|$) for a query. The idea is to assume that every relevant retrieved document is useful in satisfying $\frac{1}{p}$ th of the query. However for nonrelevant documents in set $NREL_{ij}$, a constant negative use index $(-\frac{1}{30})$ is assigned.

ii) The bottom set of documents, BOT_{ij} is not considered in the hope that REL_{ij} and $NREL_{ij}$ sets are enough in determining the use indices of documents.

Computational Complexity:

The time and space complexity of this algorithm is the same as that of algorithm S2 and is given by equations (1.5).

3. Experimental Results

Since the performance of document retirement algorithms (like most information retrieval algorithms) depends upon the unpredictable future queries, it is meaningless to talk about a proof of correctness of such algorithms. Therefore, we resort to experimental methods to evaluate the success of these algorithms.

```
Line #   Cost

 1                       procedure RETIRE comment for algorithm S3
 2                         begin
 3                           for i ← 1 step 1 until n do USENDX(i) = 0;
 4                             comment n = number of documents
 5                           for j ← 1 step 1 until BATCHNUM do
 6                             begin
 7                               for i ← 1 step 1 until BATCHSIZE do
 8                                 begin
 9                                   retrieve r documents for ith query in jth batch
10                                               (call this query $q_{ij}$);
11  ~r comps/query                   form sets $REL_{ij}$ and $NREL_{ij}$;
12                                     comment let $|REL_{ij}| = p$
12.5                                 if p ≠ 0 then
13       ↑                           for each document k in $REL_{ij}$ do
14  ~r adds/query                          USENDX(k) ← USENDX(k) + $\frac{1}{p}$;
15       |                           for each document k in $NREL_{ij}$ do
16       ↓                                 USENDX(k) ← USENDX(k) + FNREL;
17                                   save ($REL_{ij}$, $q_{ij}$) pair;
18                                 end
19                               macro (DVM);
20                             end
21                           for i ← 1 step 1 until n do
22  ~$\frac{2n}{q}$ comps/query    if (USENDX(i) ≤ CUTOFF or USENDX(i) = 0) then
23                               retire the document i;
24                       end
```

Algorithm S3

Fig. 5

The description of test environment used is as follows:

Retrieval System	:	SMART
Testing Method	:	Test and Control groups [4]
Document Collection	:	CRN4S DOCS (424 documents)
Document Cluster Collection	:	CRN4S TREE1
Query Collection	:	CRN4S QUESTS (155 queries)
a) Number of Test Queries	:	125
b) Number of Control Queries	:	30 (15 Similar + 15 Nonsimilar to Test Queries)

A) Testing Procedure

In our testing method, we compare the performance of 30 Control Queries on original document collection and on the document collection modified by retirement using 125 Test Queries. This simulates the practical on-line situation. We mention that TY [1] and BS [2] have evaluated their algorithms by comparing the performance of all 155 queries on the original document collection and on the document collection modified by retirement using the same 155 queries. They assume a fixed set of queries to be used by the users and better results are and would be obtained for this situation which is, however, not a general realistic one.

B) Choosing Parameters for Retirement Algorithms

One of the parameters to be decided for document retirement is the time span after which to retire. The retirement may be done after processing a fixed number of queries or after a fixed time. It seems that the values of these parameters depend upon usage characteristics of individual subcollections within a collection. Moreover, their optimum values would need to be determined for each subcollection experimentally. The experiments conducted here make retirement decisions after processing a batch of 5 queries for the TY algorithm and after processing the total number of 125 test queries for the others.

A discussion on choosing the other parameters follows.

a) Algorithms TY and S1

We consider a procedure for choosing positive parameters FR(>1), FN(<1) and FB(<1) used in algorithms TY and S1. Suppose initial value of USENDX assigned to each document is INIT; in case of algorithm TY, INIT is the average weight of each document assumed to be constant over all documents.

<u>Notation 3.1</u>: Let count l/m/n of a document for nonnegative integers l, m and n stand for l, m and n appearances of the document in classes REL, NREL and BOT respectively.

Then the final value of USENDX for this document is:

$$INIT * (FR)^{l} * (FN)^{m} * (FB)^{n}. \qquad (3.1)$$

<u>Definition 3.1</u>: A count $l_1/m_1/n_1$ is <u>equivalent to</u> <u>(less than)</u> a count $l_2/m_2/n_2$ of a document if the final value of USENDX obtained by $l_1/m_1/n_1$ is algebraically equal to (less than) the final value of USENDX obtained by $l_2/m_2/n_2$ i.e.

$$(FR)^{l_1} * (FN)^{m_1} * (FB)^{n_1} = (<) \; (FR)^{l_2} * (FN)^{m_2} * (FB)^{n_2}.$$

<u>Lemma 3.1</u>: Suppose that for algorithm TY without DVM and for algorithm S1 with or without DVM, a appearances of a document in class REL are offset by b appearances of the same document in class NREL or by c appearances of the same document in class BOT, i.e.:

$$a/b/0 \equiv 0/0/0 \equiv a/0/c, \qquad (3.2)$$

then

$$FN = (FR)^{-\frac{a}{b}} \quad \text{and } FB = (FR)^{-\frac{a}{c}}. \qquad (3.3)$$

Moreover, to retire all the documents with counts l/m/n or less, the retirement cutoff for USENDX may be taken as

$$CUTOFF = INIT * (FR)^{l} * (FN)^{m} * (FB)^{n} = INIT * (FR)^{l-m\cdot\frac{a}{b} - n\cdot\frac{a}{c}} \qquad (3.4)$$

<u>Proof</u>: $a/b/0 \equiv 0/0/0 \Rightarrow (FR)^{a} * (FN)^{b} = 1 \Rightarrow FN = (FR)^{-\frac{a}{b}}$

$a/0/c \equiv 0/0/0 \Rightarrow (FR)^{a} * (FB)^{c} = 1 \Rightarrow FB = (FR)^{-\frac{a}{c}}$

Rest of the lemma is obvious.

Note that the above lemma does not obviously apply to algorithm TY used with DVM since the average weight of a document also gets changed by DVM process and equation (3.1) may not hold.

Now the problem of choosing parameters FR, FN, FB and CUTOFF boils down to comparatively easy problem of choosing reasonable values of a, b, c and l, m, n and any initial values INIT and FR, both >1.

<u>Example</u>:

Let a = 1, b = 2, c = 8, i.e. 1/2/0 = 0/0/0 = 1/0/8. Also assume l = 0, m = 3, n = 0, i.e. we decide to retire all documents with counts $\leq$ 0/3/0.

Choose INIT = 12
FR = 1.56

Then equation (3.3) → $FN = (FR)^{-\frac{a}{b}} = (1.56)^{-\frac{1}{2}} = 0.8007$

and $FB = (FR)^{-\frac{a}{c}} = (1.56)^{-\frac{1}{8}} = 0.946$

$$\text{Equation (3.4)} \Rightarrow \text{CUTOFF} = \text{INIT} * (\text{FR})^{1-m\cdot\frac{a}{b} - n\cdot\frac{a}{c}}$$

$$= 12 * (1.56)^{-3\cdot\frac{1}{2}} = 6.17$$

We also note that there seems to be no straightforward way to choose parameters RELTOP, NONTOP, NONLIM and NUMBOT. However, the experiments done here tended to support that with NUMBOT = 10, the results were a little better than with NUMBOT = 0. The following seemingly reasonable values of these parameters as used by Tai and Yang were mostly used throughout experiments done here:

RELTOP = 30, NONTOP = 6, NONLIM = 5, NUMBOT = 10.

b) Algorithm S2

Here since USENDX's are changed by additions, so for a document with count l/m/n the final USENDX value is

$$\text{INIT} + (l * \text{FREL} + m * \text{FNREL} + n * \text{FBOT}) \qquad (3.5)$$

where FREL, FNREL and FBOT are parameters used here corresponding to FR, FN and FB used in algorithms TY and S1.

With this difference, the ana'ysis is similar to that done previously.

<u>Example</u>:

Choose a, b, c such that 1/5/0 ≡ 0/0/0 ≡ 1/0/20

$$\text{INIT} = 0.0$$

$$\text{FREL} = 0.5$$

Then $\text{FNREL} = -\text{FREL} * \frac{a}{b} = -0.5 * \frac{1}{5} = -0.1$

$\text{FBOT} = -\text{FREL} * \frac{a}{c} = -0.5 * \frac{1}{20} = -0.025$

To retire documents with count $\leq$ 0/0/10

CUTOFF = INIT + 10 * FBOT = 0.0 + 10* -0.025 = - 0.25

c) Algorithm S3

In this algorithm, documents appearing in BOT set are not considered. To retire documents with count $\leq$ 0/10/-, choosing a "reasonable" value of FNREL = -1/20, use

$$\text{CUTOFF} = 10 * \text{FNREL} = - 10.\frac{1}{20} = - 0.5.$$

We further note the following:

1) In addition to retiring documents with counts l/m/n or less, it is probably desirable to retire documents whose USENDX does not change at all. Such documents have either count of 0/0/0 meaning that they are either never retrieved or have some count $l_1/m_1/n_1$ suc: that

$$(\text{FR})^{l_1} * (\text{FN})^{m_1} * (\text{FB})^{n_1} = 1 \text{ for TY and S1 algorithms}$$

or (3.7)

$$l_1 * \text{FREL} + m_1 * \text{FNREL} + n_1 * \text{FBOT} = 0 \text{ for S2 and S3 algorithms.}$$

The latter category of documents may be useful although the i‿ems are still retired; however, the number of such documents is expected to be small. Since there seems to be no particular reason why this "latter category" of documents should be retired, we may modify the algorithms to prevent their retirement. There are two ways:

i) Choose FR, FN and FB (or FREL, FNREL and FBOT) in such a way that the probability of (3.7) getting satisfied for some nontrivial count $l_1/m_1/n_1$ is close to zero. For example, in (3.6) choose FNREL = $- 0.1 + \varepsilon_1$, FBOT = $- 0.025 + \varepsilon_2$ with FREL = 0.5 for some small values of ε_1 and ε_2.

ii) Depending upon the algorithm being used, choose the parameters FR > FN > FB > 1, INIT = 1 and CUTOFF > 1 (or FREL > FNREL > FBOT > 0, INIT = 0 and CUTOFF > 0). Since the USENDX of a document may only increase, (3.7) will never be satisfied for some nontrivial count $l_1/m_1/n_1$. This means that the documents retired based on the criterion of unchanged USENDX can only have count 0/0/0.*

In the experiments done here none of the above approaches is used. Thus, for instance, no particular attention is given to choosing the parameters in the sense of above approach (a). However, it is found in the experiments that about 2% documents are in this "latter category" and their retirement actually does not degrade the performance significantly.

In practice, probably the second approach should be used which has another advantage also: FREL, FNREL, FBOT and CUTOFF may be chosen to be positive integers and since the USENDX's are reinitialized after processing a group of queries, the magnitude of a USENDX would fit in a halfword. Thus the storage required for USENDX's is reduced to half.

2) If it is desired to retire a fixed % of documents say e.g. to maintain a fixed number of active documents in the store, then the retirement cutoff may be determined as follows:

*This approach was actually considered at the time the project was originally conceived. At that time it was felt that it might be desirable to use all the past information on the usefulness of a document in making future retirement decisions. So after making every retirement decision, the USENDX's of documents should not be reinitialized. In the case of the approach being considered, this means that all the USENDX's would grow without bound requiring unbounded storage. Therefore, this approach was abandoned and the algorithms were programmed as shown in the text. Presently, however, it seems that reinitializing of USENDX's after every retirement decision step may actually help in keeping the document space more up-to-date. (In practice, the optimum frequency of reinitialization of USENDX's may have to be determined experimentally.)

If d documents are to be retainea, then from the USENDX's of n documents determine the d + 1st largest USENDX by the median algorithm [5] in O(n) steps. Take this value to be CUTOFF (note some care may have to be taken to retain d documents in case dth and d + 1st largest USENDX's are the same).

This procedure takes O(n) comps/q queries or O(n/q) comps/query of O(c) comps/query for some constant c because we have assumed q = O(n). Thus, the asymptotic cost of the algorithm does not change with this technique.

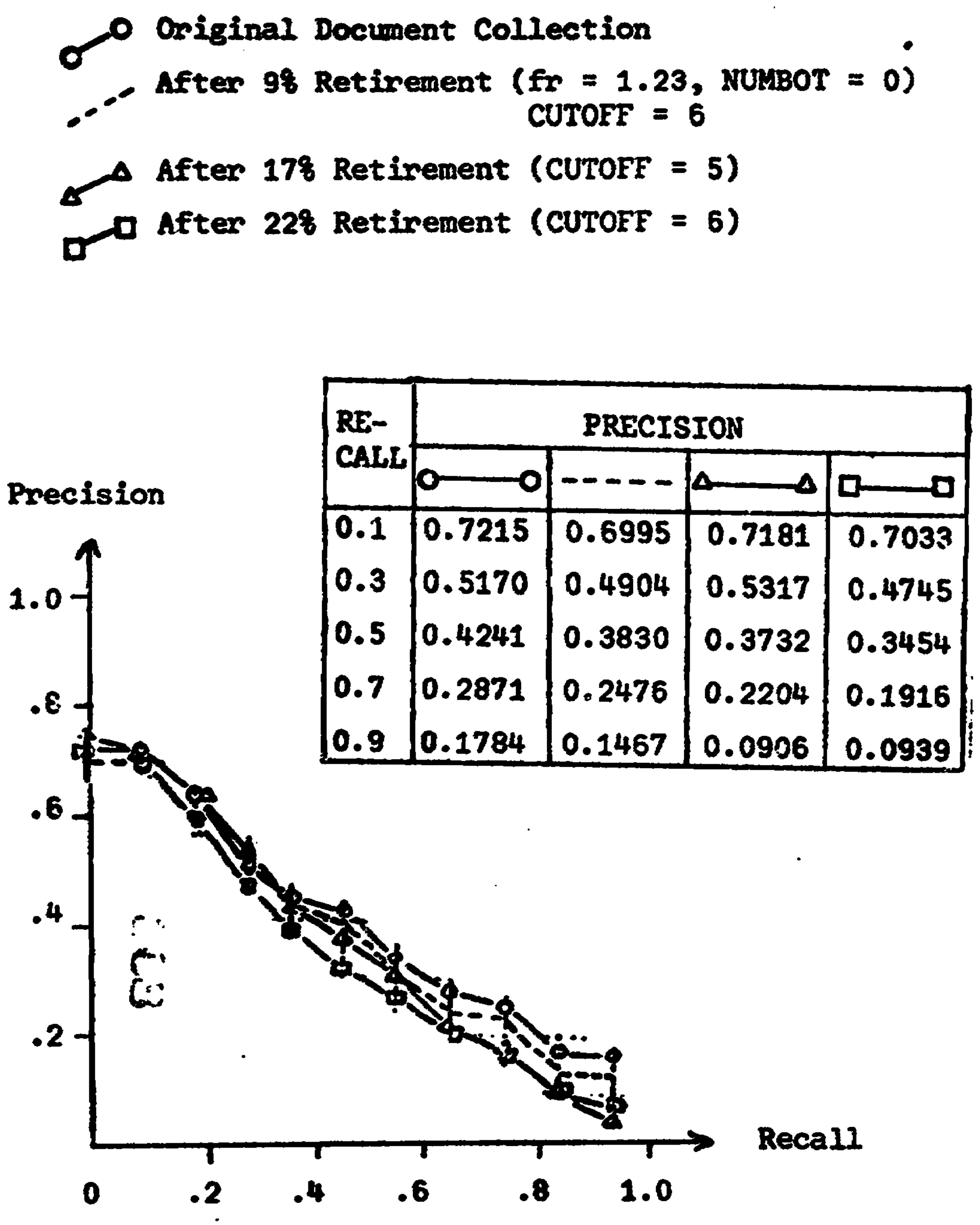
We also note that for a fixed retirement, algorithms S1 and S2 yield the same result; thus it is preferable to use the cheaper algorithm S2 in this regard.

C) P-R Curves

The Precision-Recall (P-R) curves obtained for the various algorithms without and with DVM are shown in Figs. 6-9. BS algorithm's performance curves are not obtained since this algorithm is quite costly and its results are not expected to be better than those of TY algorithm. For each algorithm the P-R curves are given for the original document collection and the collection obtained after various percentages of retirement of documents by the respective algorithms. Observations from these curves are briefly summarized below:

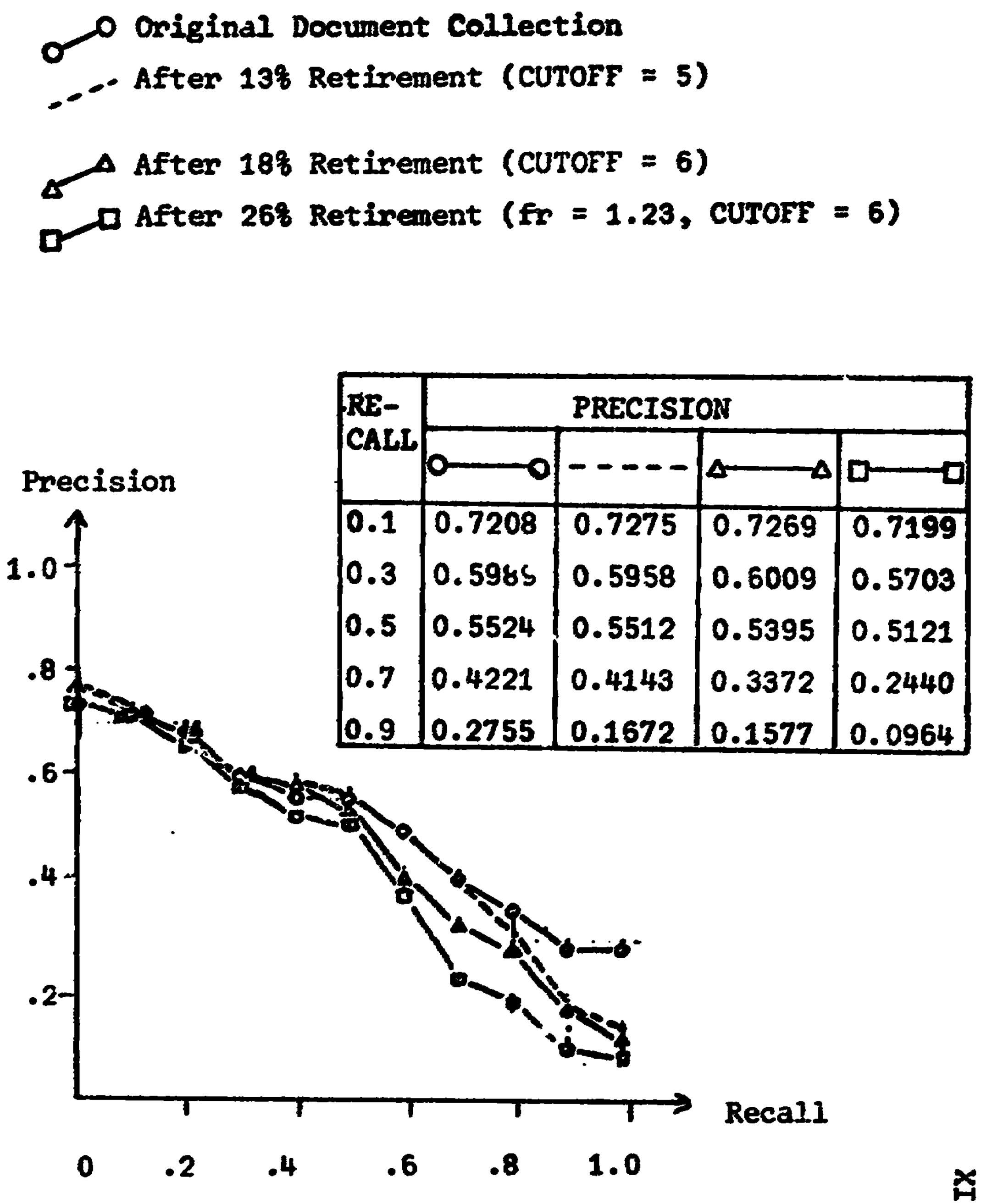
i) The performance almost monotonically degrades as the retirement rate is increased for any of the algorithms, with or without DVM.

ii) The rate of degradation of performance with the increase in retirement rate with or without DVM, is the smallest for TY algorithm upto 0.5 recall while for recall beyond 0.5 all the algorithms seem to fare equally bad. Typically, for about 18% document retirement without DVM at 0.5 recall level, the losses in precision for algorithms TY, S1, S2 and S3 are 0.050, 0.055, 0.072 and 0.070 respectively.



RE-CALL	PRECISION			
	○—○	- - - - -	△—△	□—□
0.1	0.7215	0.6995	0.7181	0.7033
0.3	0.5170	0.4904	0.5317	0.4745
0.5	0.4241	0.3830	0.3732	0.3454
0.7	0.2871	0.2476	0.2204	0.1916
0.9	0.1784	0.1467	0.0906	0.0939

a) No DVM



RE-CALL	PRECISION			
	○—○	- - - - -	△—△	□—□
0.1	0.7208	0.7275	0.7269	0.7199
0.3	0.5965	0.5958	0.6009	0.5703
0.5	0.5524	0.5512	0.5395	0.5121
0.7	0.4221	0.4143	0.3372	0.2440
0.9	0.2755	0.1672	0.1577	0.0964

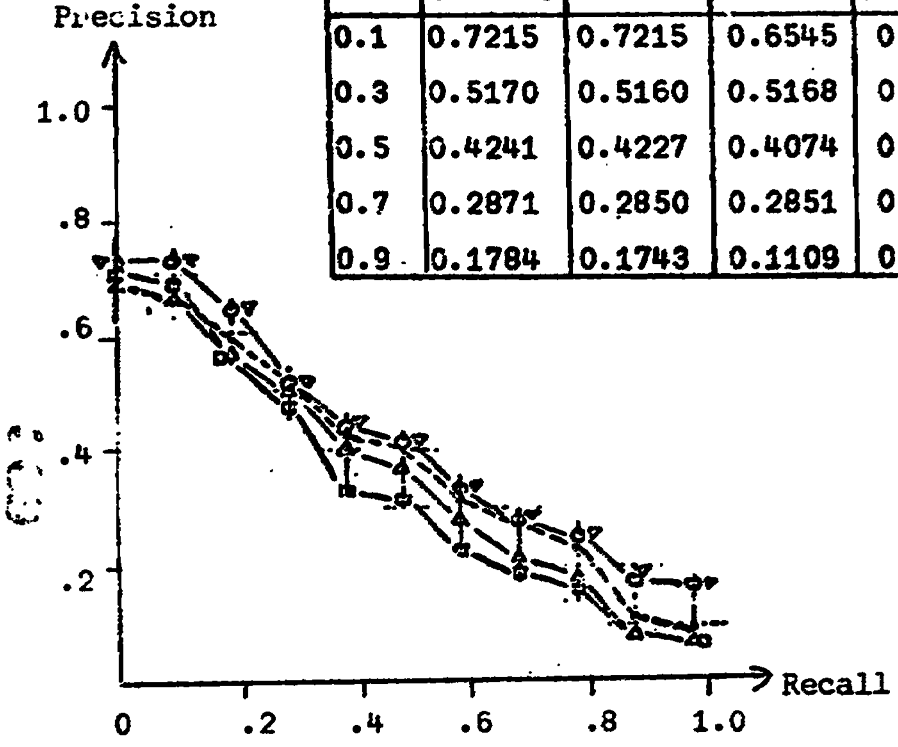
b) With DVM

Algorithm TY, Tai and Yang's Algorithm
Retirement Experiment - CRN4S DOCS Collection
Parameters: fr = 1.56, fn = 0.8, fb = 0.95, NUMBOT = 10, INIT = 12
RELTOP = 30, NONTOP = 6, NONLIM = 5 unless otherwise
explicitly stated

Fig. 6

○—○ Original Document Collection
▽—▽ After 5% Retirement (CUTOFF = 0.000)
•---• After 11% Retirement (CUTOFF = 4.435)
△—△ After 18% Retirement (CUTOFF = 5.827)
□—□ After 27% Retirement (CUTOFF = 7.284)

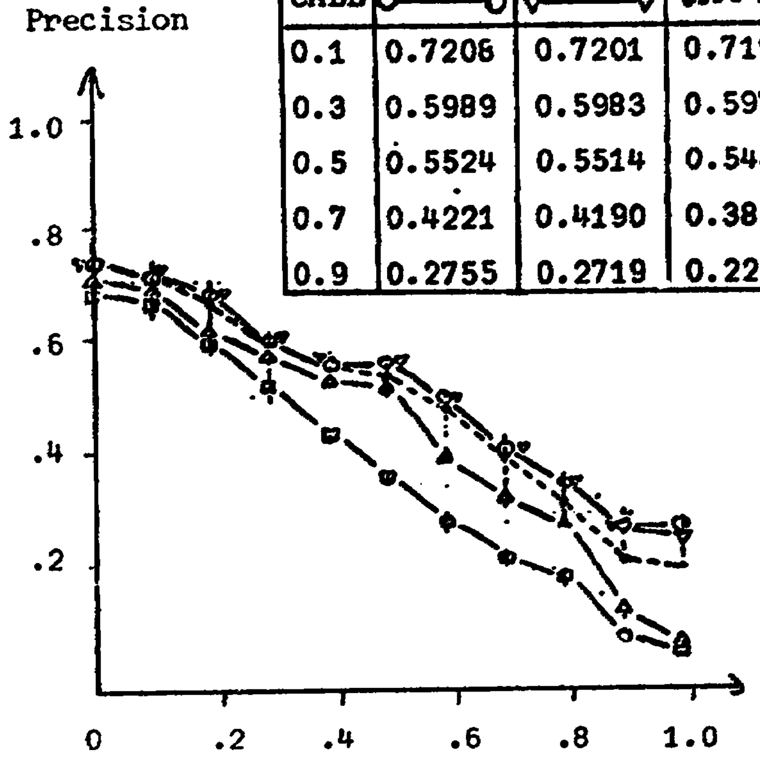
RE- CALL	PRECISION ○—○	 ▽—▽	 •---•	 △—△	 □—□
0.1	0.7215	0.7215	0.6545	0.6620	0.6848
0.3	0.5170	0.5160	0.5168	0.5045	0.4666
0.5	0.4241	0.4227	0.4074	0.3690	0.3138
0.7	0.2871	0.2850	0.2851	0.2220	0.1811
0.9	0.1784	0.1743	0.1109	0.0704	0.0726



a) No DVM

○—○ Original Document Collection
▽—▽ After 5% Retirement (CUTOFF = 0.000)
•---• After 11% Retirement (CUTOFF = 4.669)
△—△ After 18% Retirement (CUTOFF = 5.933)
□—□ After 26% Retirement (CUTOFF = 7.284)

RE- CALL	PRECISION ○—○	 ▽—▽	 •---•	 △—△	 □—□
0.1	0.7206	0.7201	0.7190	0.6885	0.6612
0.3	0.5989	0.5983	0.5973	0.5735	0.5177
0.5	0.5524	0.5514	0.5480	0.5105	0.3659
0.7	0.4221	0.4190	0.3887	0.3325	0.2129
0.9	0.2755	0.2719	0.2208	0.1347	0.0885



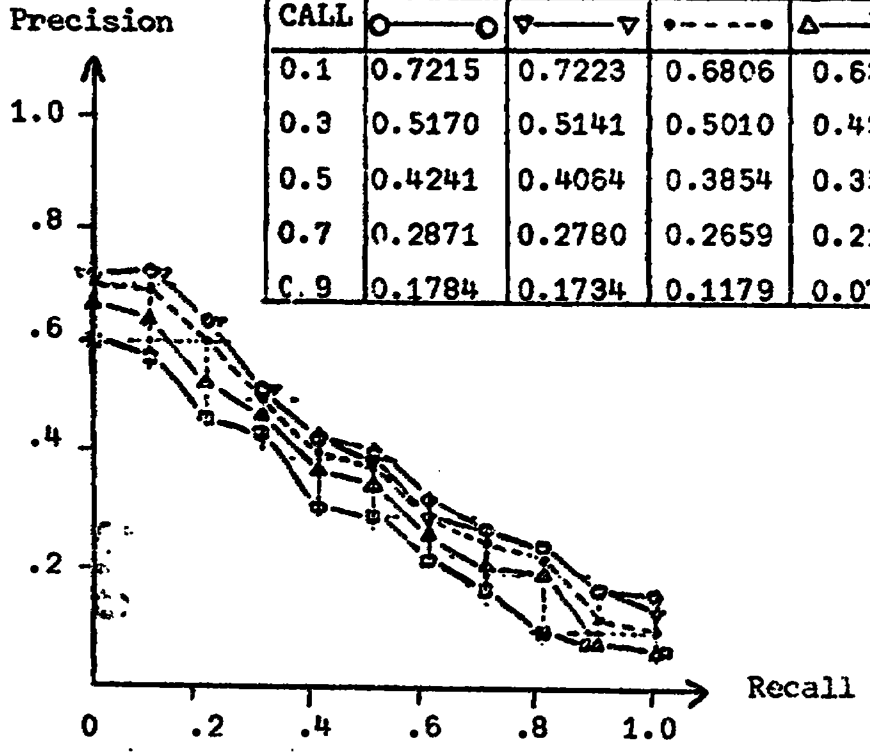
b) With DVM

Algorithm S1
Retirement Experiment - CRN4S DOCS Collection
Parameters: fr = 1.56, fn = 0.8, fb = 0.95, NUMBOT = 10, INIT = 12
RELTOP = 30, NONTOP = 6, NONLIM = 5

Fig. 7

O—O Original Document Collection
▽—▽ After 5% Retirement (CUTOFF = -1.000)
•---• After 11% Retirement (CUTOFF = -0.374)
△—△ After 19% Retirement (CUTOFF = -0.224)
□-□ After 27% Retirement (CUTOFF = -0.124)

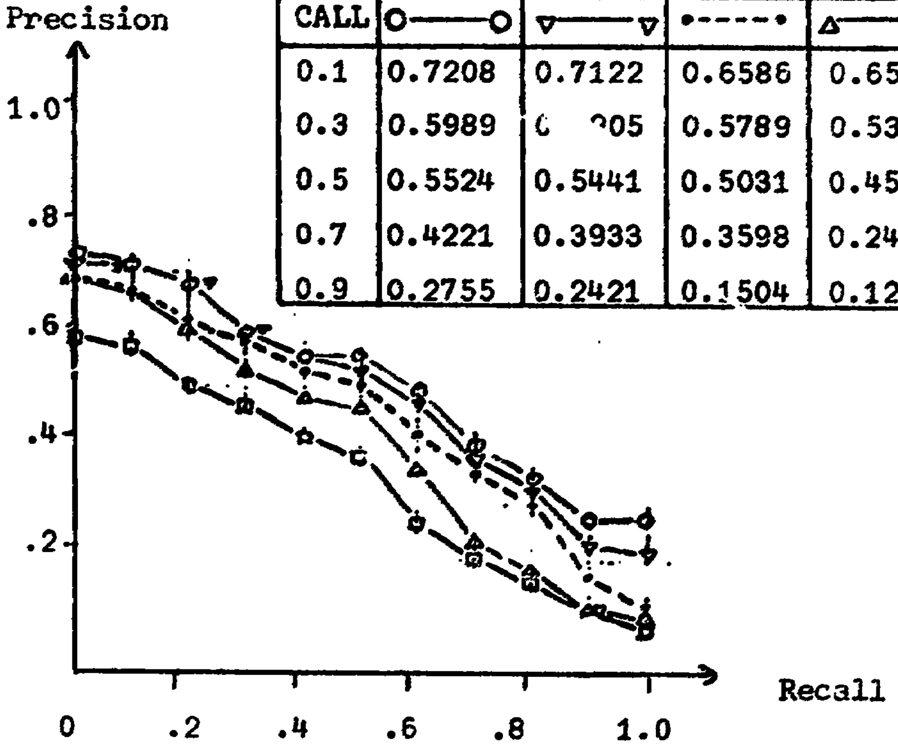
RE-CALL	PRECISION				
	O—O	▽—▽	•-----•	△—△	□—□
0.1	0.7215	0.7223	0.6806	0.6368	0.5660
0.3	0.5170	0.5141	0.5010	0.4812	0.4461
0.5	0.4241	0.4064	0.3854	0.3516	0.2988
0.7	0.2871	0.2780	0.2659	0.2190	0.1545
0.9	0.1784	0.1734	0.1179	0.0766	0.0759



a) No DVM

O—O Original Document Collection
▽—▽ After 6% Retirement (CUTOFF = -1.000)
•---• After 17% Retirement (CUTOFF = -0.230)
△—△ After 27% Retirement (CUTOFF = -0.110)
□—□ After 35% Retirement (CUTOFF = 0.000)

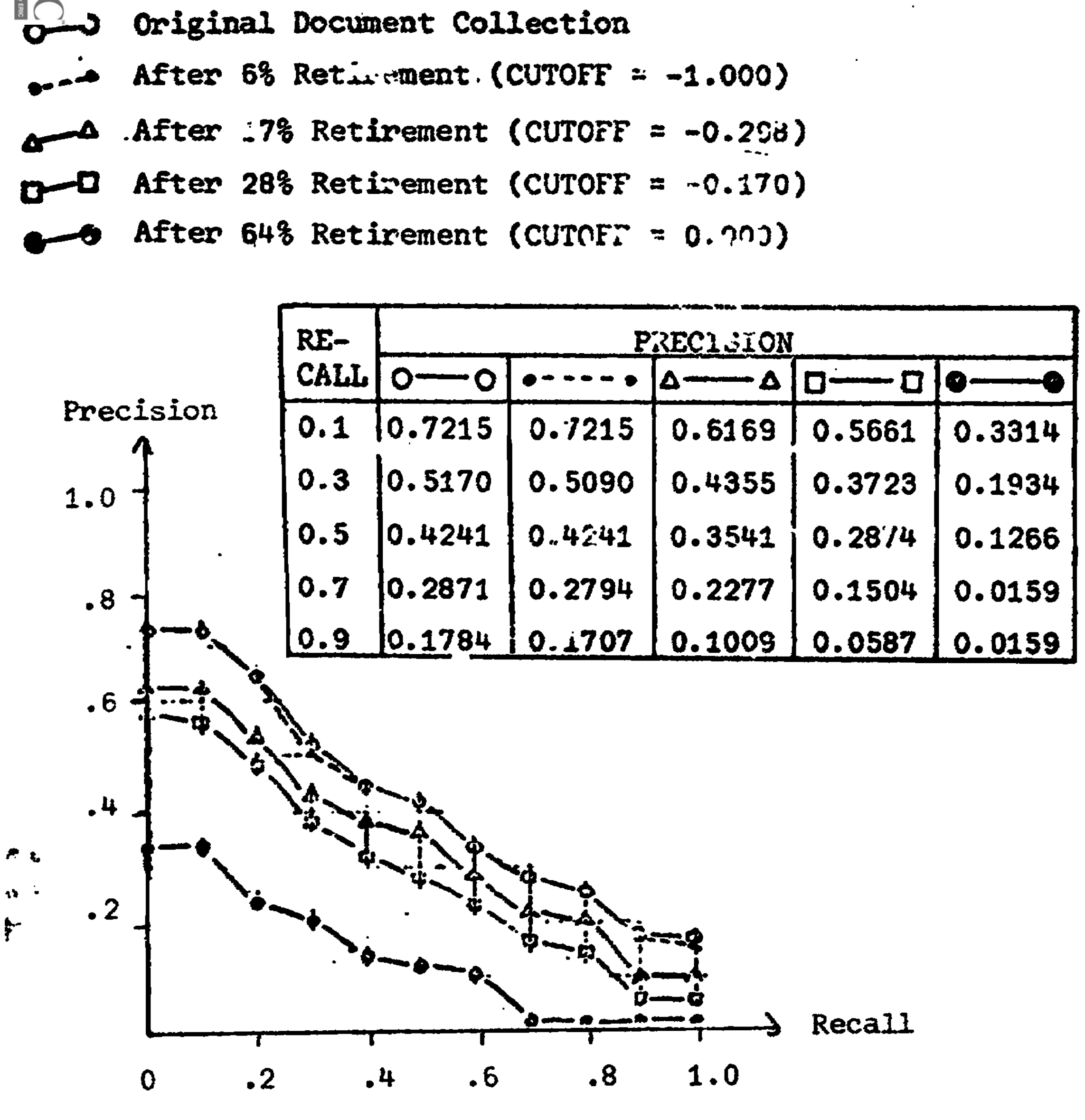
RE-CALL	PRECISION				
	O—O	▽—▽	•-----•	△—△	□—□
0.1	0.7208	0.7122	0.6586	0.6556	0.5552
0.3	0.5989	0.[illegible]05	0.5789	0.5300	0.4670
0.5	0.5524	0.5441	0.5031	0.4581	0.3878
0.7	0.4221	0.3933	0.3598	0.2439	0.2065
0.9	0.2755	0.2421	0.1504	0.1251	0.1258



b) With DVM

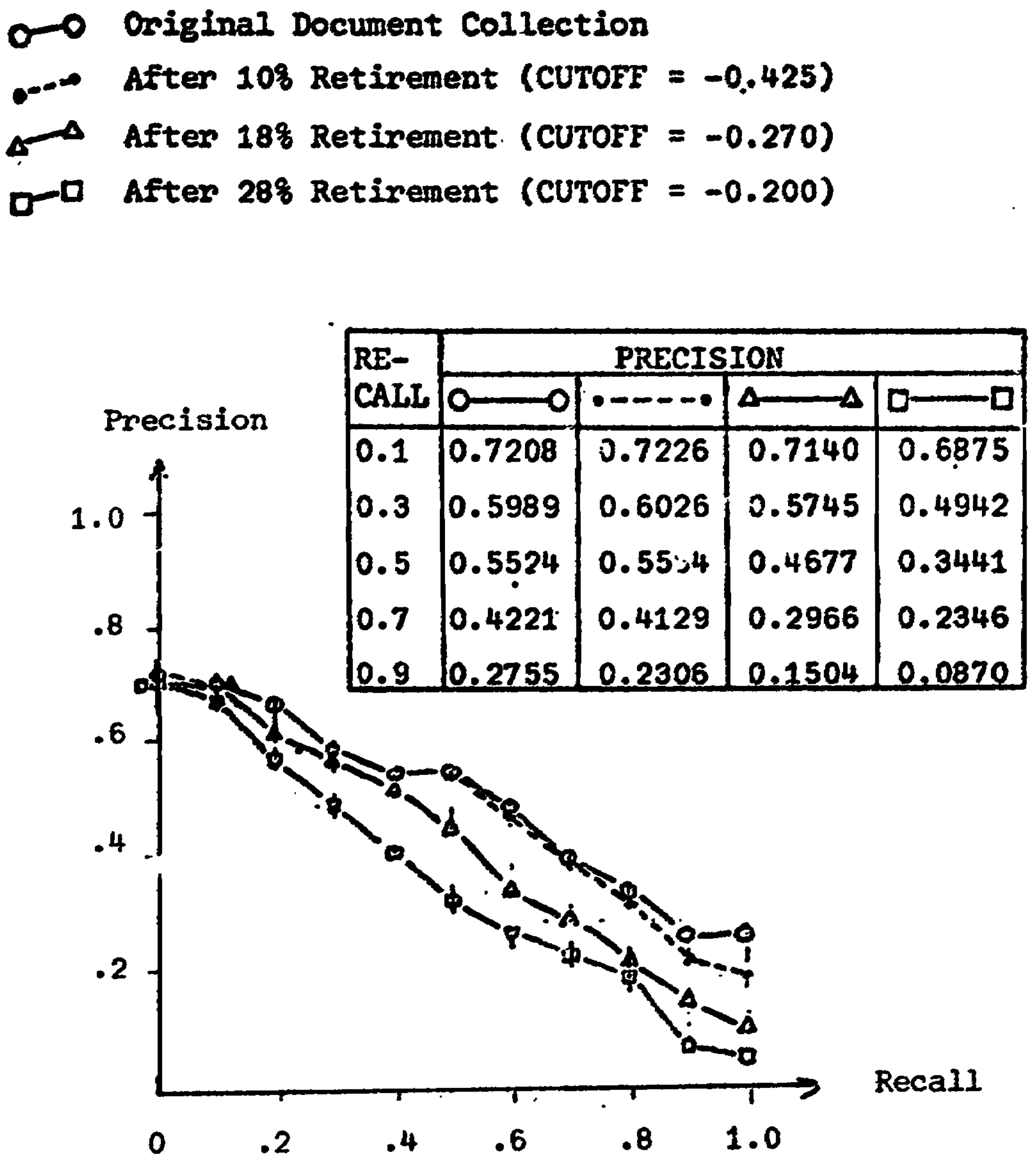
Algorithm S2
Retirement Experiment - CRN4S DOCS Collection
Parameters: FREL = 0.5, FNREL = 0.1, FBOT = -0.025, NUMBOT = 10,
INIT = 0.0, RELTOP = 30, NONTOP = 6, NONLIM = 5

Fig. 8



RE-CALL	PRECISION				
	○——○	•- - - -•	△——△	□——□	●——●
0.1	0.7215	0.7215	0.6169	0.5661	0.3314
0.3	0.5170	0.5090	0.4355	0.3723	0.1934
0.5	0.4241	0.4241	0.3541	0.2874	0.1266
0.7	0.2871	0.2794	0.2277	0.1504	0.0159
0.9	0.1784	0.1707	0.1009	0.0587	0.0159

a) No DVM



RE-CALL	PRECISION			
	○——○	•- - - -•	△——△	□——□
0.1	0.7208	0.7226	0.7140	0.6875
0.3	0.5989	0.6026	0.5745	0.4942
0.5	0.5524	0.5534	0.4677	0.3441
0.7	0.4221	0.4129	0.2966	0.2346
0.9	0.2755	0.2306	0.1504	0.0870

b) With DVM (Relevant documents divided into 3 classes, top 10, middle 10 and bottom 10 and assigned relative weight 3, 2 and 1 respectively, the weighted contribution of each document is used)

Algorithm S3
Retirement Experiment - CRN4S DOCS Collection
Parameters: FNREL = 1/20, RELTOP = 30,
NONTOP = 20, NONLIM = 20, INIT = 0.00

Fig. 9

iii) The only difference in retirement methods with and without DVM is in that all the P-R curves with DVM are higher than the corresponding curves without DVM but relative to each individual group, the performance is similar in both the cases.

iv) For retirement upto 10% or so, the performance is much like the one with the original collection for TY algorithm. The same is true for other algorithms upto retirement of about 6%. However, the retirement of 6% or so for algorithms S1, S2 and S3 is obtained by retiring only the documents whose USENDX remains stationary after processing 125 queries since CUTOFF values of 0.000 for S1 algorithm and of -1.000 for S2 and S3 algorithms were so chosen that no USENDX of documents would be below these values.

This means that for collections like CRN1400 which contains around 30% documents not relevant to any query, since USENDX for such documents has a high probability of remaining invariant, such idle documents would be retired by either of the algorithms S1, S2 or S3 without affecting the retrieval performance. The same may not hold for TY algorithm.

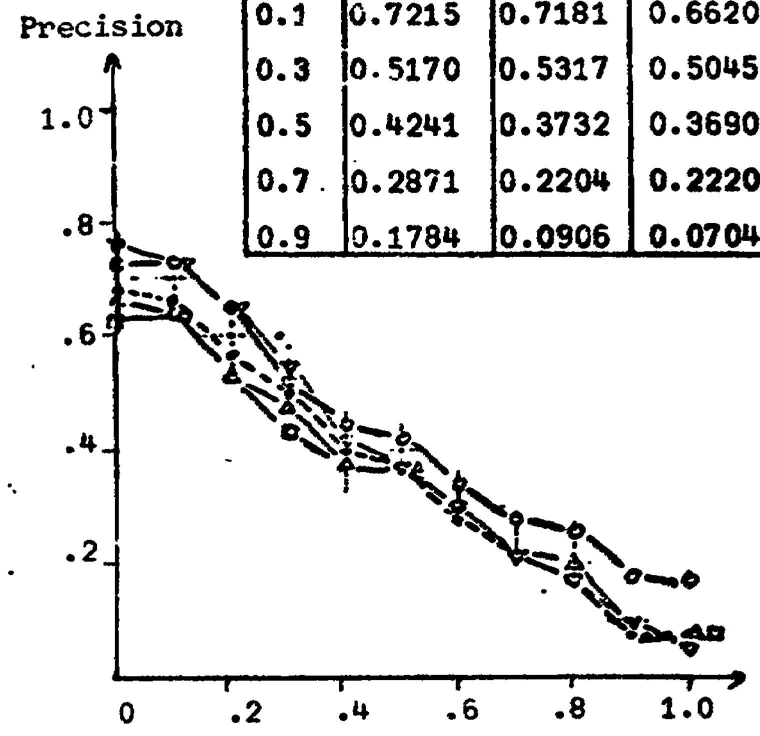
v) Only S3 algorithm was tried for document retirement as high as 64% and that too only without DVM (Fig. 9(a)). The performance gets progressively worse compared to the performance of the original collection. It is then expected that same would be true of other algorithms also even though it is not very clear as to by how much the TY algorithm will deteriorate at such high retirement rate.

4. Overall Comparison of Various Algorithms

Figs. 10 and 11 give the overall comparison of various algorithms. Fig. 10 gives the sample comparison between the various algorithms - at 17-19% retirement without DVM and at 26-28% retirement with DVM. It is found that

○ Original Document Collection
▽—▽ After 17% Retirement (Algorithm TY)
•---• After 18% Retirement (Algorithm S1)
△—△ After 19% Retirement (Algorithm S2)
□—□ After 17% Retirement (Algorithm S3)

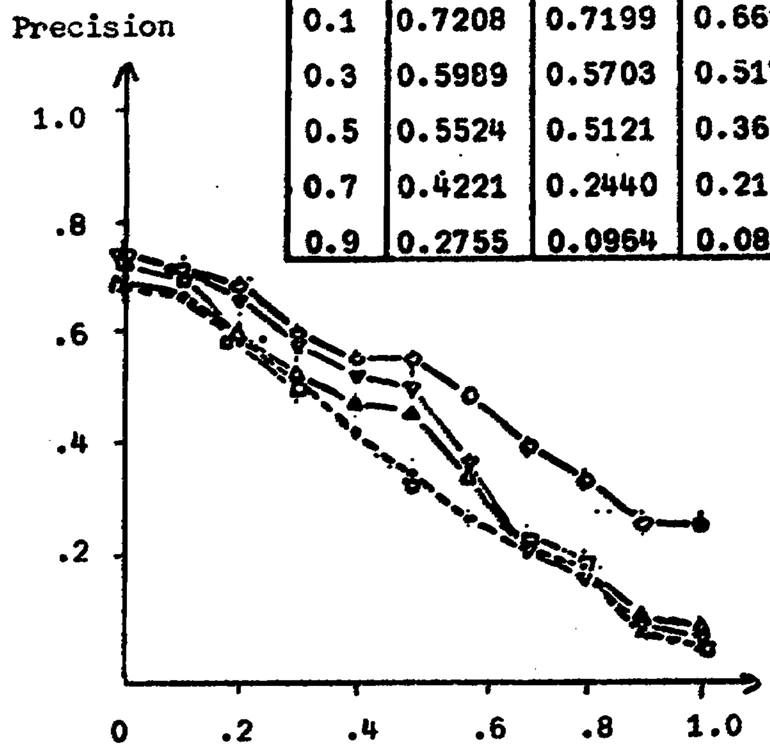
RE- CALL	PRECISION ○—○	▽—▽	•----•	△—△	□—□
0.1	0.7215	0.7181	0.6620	0.6368	0.6169
0.3	0.5170	0.5317	0.5045	0.4812	0.4355
0.5	0.4241	0.3732	0.3690	0.3516	0.3541
0.7	0.2871	0.2204	0.2220	0.2190	0.2277
0.9	0.1784	0.0906	0.0704	0.0765	0.1009



a) No DVM

○—○ Original Document Collection
▽—▽ After 26% Retirement (Algorithm TY)
•---• After 26% Retirement (Algorithm S1)
△—△ After 27% Retirement (Algorithm S2)
□—□ After 28% Retirement (Algorithm S3 with some modification as given in Fig. 9)

RE- CALL	PRECISION ○—○	▽—▽	•----•	△—△	□—□
0.1	0.7208	0.7199	0.6612	0.6556	0.6875
0.3	0.5989	0.5703	0.5177	0.5300	0.4942
0.5	0.5524	0.5121	0.3659	0.4581	0.3441
0.7	0.4221	0.2440	0.2129	0.2439	0.2346
0.9	0.2755	0.0964	0.0885	0.1251	0.0870



b) With DVM

Fig. 10 Comparison of Various Algorithms
Retirement Experiment - CRN4S DOCS Collection
Comparison made for 17-19% Retirement without DVM
and 26-28% Retirement with DVM

Algorithm Compared	RETIREMENT %												Running Time of the Algorithm per query
	5	6	7	10	11	16	17	18	20	25	26	28	
S1	~o				~TY			≲TY			~TY		O(r)
S2	~o				~TY				~TY		<TY		O(r)
S3		~o					~TY					≲TY	O(r)

A) No DVM

Algorithm Compared	RETIREMENT %												Running Time of the Algorithm per c y
	5	6	7	10	11	16	17	18	20	25	26	28	
S1	~o				>TY			≲TY		<TY			O(r)
S2		>TY			~TY	≲TY					<TY		O(r)
S3 modified as shown in Fig. 9			~o	≲o				<TY				<TY	O(r)

B) With DVM

Note: Running time of algorithms TY and BS is O(mr) per query

Overall good algorithms (in order) ⟵ S3, S2, S1 ⟶⟵ S1, S2, S3 ⟶⟵ S1, S2, S3 ⟶

comments: ~o | ~TY | ~, ≲ or <TY

o - Original, ∴ - Tai and Yang Algorithm, ≲ - Just a little worse, ~ - Almost equivalent, > - Better than

Comparison of Document Retirement Algorithms, S1 S2, and S3 with TY
Fig. 11

most of the curves are clustered together beyond 0.5 recall while some differences are apparent at low recall especially from the inserted recall-precision tables. Except for retirement of less than 10% when all the algorithms give performance close to the original, TY algorithm seems to give overall better P-R curves. For this reason and because the running time of TY algorithm is O(mr) operations/query compared to O(r) operations/query for algorithms S1, S2 and S3, the latter algorithms are compared with TY algorithm closely at different retirement rates.

The results are tabulated in Fig. 11 for with and without DSM cases separately. The following observations are made:

i) Upto about 7% retirement, algorithm S3, S2 and S1 give results equivalent to the original document collection and/or better than TY algorithm.

ii) For 8 - 17% retirement algorithms S1, S2 and S3 give performance equivalent to or a little worse than TY algorithm.

iii) For 18 - 28% retirement, algorithms S1, S2 and S3 give performance starting from equivalent to worse than TY algorithm.

iv) Algorithm S1 seems to do overall better than S2 and S3 algorithms.

Considering that algorithms S1, S2 and S3 cost O(r) operations/query compared to O(mr) operations/query for TY algorithm and that the former algorithms give performance almost equivalent to the latter especially for high recall, a safe conclusion may be made that cheaper algorithms should be used for document retirement.

5. Summary and Conclusions

This paper has attempted to study the behavior of various automatic document retirement algorithms including the previously known ones and some new proposed algorithms. The cost of using these algorithms is analyzed and the effects of these algorithms on the storage costs and performance of the retrieval system are examined. The conclusions are summarized below:

i) Without cost consideration, algorithm TY probably is the best of the algorithms tried. This algorithm costs $O(mr)$ operations/query.

ii) Allocating a computer word or so for USENDX for each document i.e. at the expense of $O(n)$ additional space (this is, however, cheap off device storage), the running cost using algorithm S1 is $O(r)$ operations/query with performance very much like algorithm TY especially at high recall. Algorithm S1 is m (on the average m = 200) times faster than algorithm TY in running time.

iii) The performance of algorithm S1 is like algorithm TY for upto about 18% retirement and even then S1 gets worse for low recalls (≤ 0.5) mostly.

iv) With algorithms TY and S1, the performance with upto 13% retirement is pretty close to the performance of the original document collection.

v) Even with algorithms S2 and S3 which require $O(n)$ additional off-line space for USENDX's of documents and cost the least in time complexity (i.e. ~r comparisons/query), the performance with upto 10 - 12% retirement is very close to the original performance.

vi) With algorithms S1, S2 and S3, a retirement of 5 - 10% that retires all the documents whose USENDX's do not change after processing q (~125 or so) queries keeps the system performance almost exactly equal to the original performance. Note: as said at the end of Section 2(b), such retirement surely retires documents which never get retrieved nor end up in BOT set.

The documents which are not relevant to any queries also have a very good chance of retirement.

Implementation of retiring such documents with TY algorithm is difficult since initial AVGWT of each document is different and even if these values are calculated for each document, they need to be stored and this requires 0(n) space which the TY algorithm mainly tries to avoid.

vii) In view of the previous point, it is expected that with document collections like CRN1400, which contains almost 30% documents not relevant to any queries at all, the algorithms S1, S2 and S3 will retire at least these 30% documents or so, without any loss in performance. (In practice, the numb of such etired documents would be less than 30% since some of these documents will have counts $l_1/m_1/n_1$ such that Final USENDX ≠ INIT.) This may not be true of TY algorithm. In such situations, algorithms S1, S2 and S3 are expected to perform better than TY algorithm.

viii) For all the algorithms tried, the performance gets almost monontonically worse with the increase in the retirement rate.

ix) The BS algorithm, which is not tried, is expected to perform no better than TY algorithm while it is the most expensive (in time complexity) of the algorithms considered.

x) The rate of degradation of performance with the increase in retirement rate, with or without DVM, is the smallest for TY algorithm upto 0.5 recall while for recall beyond 0.5 all the algorithms seem to fare equally bad.

ix) The only difference in retirement methods with and without DSM is in that all the performance curves with DVM are higher than the corresponding curves without DVM. However, relative to each individual group, the performance is similar in both the cases.

xii) Overall, the kinds of algorithms to be used against the possible percentages of retirement may be stated as follows:

Retirement %	Algorithm Recommended (in order)
5-10%	S3, S2, S1, TY
11-18%	S2, S3, S1, TY
18-26%	S1, S2, S3, TY

xiii) To summarize in one sentence, the main conclusion of this work is that various document retirement algorithms obtain almost equivalent performance, especially at high recall; so it is preferable to use cheaper algorithms.

6. Suggestions for Further Research

As a result of the present work, the following questions, which are still unanswered, might be well worth looking into:

i) It is not quite clear why the tried version of TY algorithm, which does not retire the documents with unchanged AVGWT but does make retirement decisions after every batch of queries, performs better than other algorithms, especially its close variant - S1 algorithm, at low recall. The main suspected reason is that in TY algorithm, the retirement decisions are made more frequently (the cost obviously increases with this frequency) which keeps the document space more up-to-date. This implies the need for determining the optimum time span between successive retirement decisions, with tradeoffs between cost and performance. It seems that S1 algorithm which makes retirement decisions after this optimum time span should give the overall best performance.

ii) What should be optimum frequency of reinitialization of USENDX's of documents? How does it affect the cost and performance of the system? This may have to be determined experimentally in the practical environment being considered.

iii) How will random retirement of documents behave compared to other algorithms?

iv) How will the relative results of different algorithms vary with the use of different collections? In addition, it would be nice to see some research done to resolve the following questions.

v) How can the ideas of obsolescence of literature based on statistical analysis of the collections, as used by Brookes [6], etc. be combined with the document retirement based on relevancy decisions? We feel that the analysis of this question would require the use of collections larger than the ones presently available for the SMART system.

vi) How will the retirement and/or the promotion of documents between different levels of storage affect the cost of transportation of documents, cost of modifying the clusters and/or the centroids, optimum reorganization points for the data base, cost of modifying the thesaurus, stability of the document space, etc.? How will the retrieval performance which also considers the system efficiency factor (not considered in this report) i.e. the time required to search through the hierarchical data base to find the desired information, behave as a result of these?

vii) How much of the work done and ideas used in paging algorithms of operating systems may be useful in the area of document retirement?

viii) Lastly, it might be nice to see all the ideas of document retirement put together into making a viable model and theory of document retirement.

References

[1] Tai, K. and C.S. Yang, "Document Retirement in Automatic Retrieval Systems", Report ISR-21 to the National Library of Medicine, Section VII, Department of Computer Science, Cornell University, Ithaca, N.Y., December 1972.

[2] Beall, J.C. and M.D. Schnitzer, "A Method for Automatic Document Retirement", CS 435 Project Report, Department of Computer Science, Cornell University, Ithaca, N.Y., Spring 1973.

[3] Aho, A.V., J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley Publishing Co., 1974.

[4] Brauen, T.L. "Document Vector Modification", Chapter 24 in The SMART Retrieval System — Experiments in Automatic Document Processing, G. Salton editor, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[5] Blum, M., R.E. Floyd, V. Pratt, R.L. Rivest and R.E. Tarjan, "Linear Time Bounds for Median Computations", Conference Record of the Fourth ACM Symposium on Theory of Computing, 1972, pp. 119-124.

[6] Brookes, B.C., "Optimum P% Library of Scientific Periodicals", Nature, Vol. 232, August 13, 1971, pp. 458-461.