DOCUMENT RESUME

ED 048 914 LI 002 723

TITLE	Clustering Methods; Part IV of Scientific Report No. ISR-18, Information Storage and Retrieval...
INSTITUTION	Cornell Univ., Ithaca, N.Y. Dept. of Computer Science.
SPONS AGENCY	National Library of Medicine (DHEW), Bethesda, Md.; National Science Foundation, Washington, D.C.
REPORT NO	ISR-18 [Part IV]
PUB DATE	Oct 70
NOTE	95p.; Part of LI 002 719
EDRS PRICE	EDRS Price MF-$0.65 HC-$3.29
DESCRIPTORS	*Algorithms, Automation, *Classification, *Cluster Grouping, Evaluation Methods, *Information Retrieval, *Search Strategies
IDENTIFIERS	On Line Retrieval Systems, *Saltons Magical Automatic Retriever of Texts, SMART

ABSTRACT

Two papers are included as Part Four of this report on Salton's Magical Automatic Retriever of Texts (SMART) project report. The first paper: "A Controlled Single Pass Classification Algorithm with Application to Multilevel Clustering" by D. B. Johnson and J. M. Laferente presents a single pass clustering method which compares favorably with more expensive clustering algorithms. The method is tested using the ADI collection of 82 documents and the Cranfield 424 Collection. The results are compared to full search and to results obtained by searching clusters produced by Dattola's algorithm. The second paper: "A Systematic Study of Query-Clustering Techniques: A Progress Report" by S. Worona describes an experiment using various techniques of query clustering on the Cranfield 424 document collection and gives some preliminary results. Several methods of evaluating the performance of clustered searches in the context of query-clustering are discussed. Some observations are also made concerning use of the SMART system as implemented at Cornell University. (For the entire SMART project report see LI 002 719, for parts 1-3 see LI 002 720 through LI 002 722, for part 5 see LI 002 724.) (NH)

Department of Computer Science

Cornell University

Ithaca, New York 14850

Clustering Methods
Part III
of

Scientific Report No. ISR-18

INFORMATION STORAGE AND RETRIEVAL

to

The National Science Foundation

and to

The National Library of Medicine

Reports on Analysis, Dictionary Construction, User Feedback, Clustering, and On-Line Retrieval

Ithaca, New York
October 1970

Gerard Salton
Project Director

SMART Project Staff

Robert Crawford
Barbara Galaska
Eileen Gudat
Marcia Kerchner
Ellen Lundell
Robert Peck
Jacob Razon
Gerard Salton
Donna Williamson
Robert Williamson
Steven Worona
Joel Zumoff

ERIC User Please Note:

This Table of Contents outlines all 5 parts of Information Storage and Retrieval (ISR-18), which is available in its entirety as LI 002 719. Only the papers from Part Four are reproduced here as LI 002 723. See LI 002 720 thru LI 002 722 for Parts 1 -3 and LI 002 724 for Part 5.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

PART TWO

AUTOMATIC DICTIONARY CONSTRUCTION

Available as LI 002 721

TABLE OF CONTENTS (continued)

Page

V. continued

"The Effect of Common Words and Synonyms on Retrieval Performance"

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

Page

PART THREE

USER FEEDBACK PROCEDURES

Available as LI 002 722

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

Page

PART FOUR

CLUSTERING METHODS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

Summary

The present report is the eighteenth in a series describing research in automatic information storage and retrieval conducted by the Department of Computer Science at Cornell University. The report covering work carried out by the SMART project for approximately one year (summer 1969 to summer 1970) is separated into five parts: automatic content analysis (Sections I to IV), automatic dictionary construction (Sections V to VII), user feedback procedures (Sections VIII to XI), document and query clustering methods (Sections XII and XIII), and SMART systems design for on-line operations (Sections XIV and XV).

Most recipients of SMART project reports will experience a gap in the series of scientific reports received to date. Report ISR-17, consisting of a master's thesis by Thomas Brauen entitled "Document Vector Modification in On-line Information Retrieval Systems" was prepared for limited distribution during the fall of 1969. Report ISR-17 is available from the National Technical Information Service in Springfield, Virginia 22151, under order number PB 186-135.

The SMART system continues to operate in a batch processing mode on the IBM 360 model 65 system at Cornell University. The standard processing mode is eventually to be replaced by an on-line system using time-shared console devices for input and output. The overall design for such an on-line version of SMART has been completed, and is described in Section XIV of the present report. While awaiting the time-sharing implementation of the system, new retrieval experiments have been performed using larger document collections within the existing system. Attempts to compare the performance

of several collections of different sizes must take into account the collection "generality". A study of this problem is made in Section II of the present report. Of special interest may also be the new procedures for the automatic recognition of "common" words in English texts (Section VI), and the automatic construction of thesauruses and dictionaries for use in an automatic language analysis system (Section VII). Finally, a new inexpensive method of document classification and term grouping is described and evaluated in Section XII of the present report.

Sections I to IV cover experiments in automatic content analysis and automatic indexing. Section I by S. F. Weiss contains the results of experiments, using statistical and syntactic procedures for the automatic recognition of phrases in written texts. It is shown once again that because of the relative heterogeneity of most document collections, and the sparseness of the document space, phrases are not normally needed for content identification.

In Section II by G. Salton, the "generality" problem is examined which arises when two or more distinct collections are compared in a retrieval environment. It is shown that proportionately fewer nonrelevant items tend to be retrieved when larger collections (of low generality) are used, than when small, high generality collections serve for evaluation purposes. The systems viewpoint thus normally favors the larger, low generality output, whereas the user viewpoint prefers the performance of the smaller collection.

The effectiveness of bibliographic citations for content analysis purposes is examined in Section III by G. Salton. It is shown that in some situations when the citation space is reasonably dense, the use of citations attached to documents is even more effective than the use of standard keywords or descriptors. In any case, citations should be added to the normal descriptors whenever they happen to be available.

In the last section of Part 1, certain template analysis methods are applied to the automatic resolution of ambiguous constructions (Section IV by S. F. Weiss). It is shown that a set of contextual rules can be constructed by a semi-automatic learning process, which will eventually lead to an automatic recognition of over ninety percent of the existing textual ambiguities.

Part 2, consisting of Sections V, VI and VII covers procedures for the automatic construction of dictionaries and thesauruses useful in text analysis systems. In Section V by D. Bergmark it is shown that word stem methods using large common word lists are more effective in an information retrieval environment that some manually constructed thesauruses, even though the latter also include synonym recognition facilities.

A new model for the automatic determination of "common" words (which are not to be used for content identification) is proposed and evaluated in Section VI by K. Bonwit and J. Aste-Tonsmann. The resulting process can be incorporated into fully automatic dictionary construction systems. The complete thesaurus construction problem is reviewed in Section VII by G. Salton, and the effectiveness of a variety of automatic dictionaries is evaluated.

Part 3, consisting of Sections VIII through XI, deals with a number of refinements of the normal relevance feedback process which has been examined in a number of previous reports in this series. In Section VIII by T. P. Baker, a query splitting process is evaluated in which input queries are split into two or more parts during feedback whenever the relevant documents identified by the user are separated by one or more nonrelevant ones.

The effectiveness of relevance feedback techniques in an environment of variable generality is examined in Section IX by B. Capps and M. Yin. It is shown that some of the feedback techniques are equally applicable to collections of small and large generality. Techniques of negative feedback (when no relevant items are identified by the users, but only nonrelevant ones) are considered in Section X by M. Kerchner. It is shown that a number of selective negative techniques, in which only certain specific concepts are actually modified during the feedback process, bring good improvements in retrieval effectiveness over the standard nonselective methods.

Finally, a new feedback methodology in which a number of documents jointly identified as relevant to earlier queries are used as a set for relevance feedback purposes is proposed and evaluated in Section XI by L. Paavola.

Two new clustering techniques are examined in Part 3 of this report, consisting of Sections XII and XIII. A controlled, inexpensive, single-pass clustering algorithm is described and evaluated in Section XII by D. B. Johnson and J. M. Lafuente. In this clustering method, each document is examined only once, and the procedure is shown to be equivalent in certain circumstances to other more demanding clustering procedures.

The query clustering process, in which query groups are used to define the information search strategy is studied in Section XIII by S. Worona. A variety of parameter values is evaluated in a retrieval environment to be used for cluster generation, centroid definition, and final search strategy.

The last part, number five, consisting of Sections XIV and XV, covers the design of on-line information retrieval systems. A new SMART system design for on-line use is proposed in Section XIV by D. and R. Williamson, based on the concepts of pseudo-batching and the interaction of a cycling program with a console monitor. The user interface and conversational facilities are also described.

A template analysis technique is used in Section XV by S. F. Weiss for the implementation of conversational retrieval systems used in a time-sharing environment. The effectiveness of the method is discussed, as well as its implementation in a retrieval situation.

Additional automatic content analysis and search procedures used with the SMART system are described in several previous reports in this series, including notably reports ISR-11 to ISR-16 published between 1966 and 1969. These reports are all available from the National Technical Information Service in Springfield, Virginia.

G. Salton

XII. A Controlled Single Pass Classification Algorithm with Application to Multilevel Clustering

D. B. Johnson and J. M. Lafuente

Abstract

A single pass clustering method is presented, which compares favorably with more expensive clustering algoritms. During the clustering process various parameters are controlled, such as number of clusters, size of the clusters and amount of overlap. The method is tested using the ADI collection of 82 documents and the Cranfield 424 collection. The results are compared to full search and to results obtained by searching clusters produced by Dattola's algorithm. The effect of ordering of the collection is investigated and some variation is obtained in the results for different orderings. Single-level as well as two-level clustering is considered. The results, in general, point to better performance with multilevel clustering and some suggestions for extending the algorithm to include multilevel clustering are given.

1. Introduction

An important consideration in an automatic information retrieval system is the time spent in searching a collection. To avoid searching the entire collection, it becomes necessary to classify documents into related groups. This is the technique of clustering. Documents are grouped into clusters by assigning items containing similar concepts to the same cluster. A centroid vector is constructed for each cluster, and queries are matched at first against these centroids. Only those clusters comparing favorably with a query are then searched in the normal manner. Thus a sacrifice in time to produce the clusters is compensated by later savings in the search time. The problem is how to develop efficient techniques for producing meaningful clusters in order to minimize searching costs. The suitability of any clustering method must then be measured according to the following criteria:

1. Cost of generating the clusters;
2. Cost of searching the clustered collection;
3. Effectiveness of the search process, usually measured by evaluating recall and precision.

The clustering problem becomes critical with very large collections. Comparing each document with every other document is no longer feasible, and efficient algorithms have been developed which attempt to minimize the number of document comparisons. Even with these methods, the classification of a collection containing several thousand items is a time consuming process.

This paper describes a clustering algorithm which makes a single pass over a collection. Each document is examined only once and clusters are formed in the process. A document is considered for inclusion into one or more existing clusters before it is allowed to begin a cluster of its own. Various parameters such as cluster size, number of clusters, and amount of overlap are controlled throughout the clustering process. The algorithm is tested using the ADI collection of 82 documents and 35 queries available in the SMART system. The algorithm, however, is designed with a view toward large collections.

2. Methods of Clustering

Various methods have been devised for clustering. Usually these methods require the computation of a correlation matrix, representing the correlation of each document with every other document in the collection, followed by a grouping of those documents which correlate best with each other. Input parameters for these clustering algorithms include the number of clusters desired, the maximum size of each cluster, the amount of overlap and the number of "loose" or unclassified documents to be allowed.

Two clustering methods are presently in use in the SMART System; [1] they are: Rocchio's clustering algorithm [2] and a variation of Doyle's algorithm. [3] In Rocchio's algorithm, each unclustered document is selected as a candidate for a cluster nucleus. All remaining documents are then correlated with it, and the document is subjected to a density test based on cut-off correlation coefficients. If the document passes the density test, a new cluster is formed and a cut-off correlation is determined based on the relative distribution of correlations with the given document. A centroid vector is then computed by combining all concepts of those documents with correlation above the computed cut-off value. The centroid vector is next matched against the entire collection to create an altered cluster. The entire procedure is now repeated with all unclustered documents until all documents are either clustered or loose.

Doyle's algorithm basically consists in matching documents to existing clusters by computing a document-cluster score for each document relative to each cluster and admitting a document to those clusters for which a sufficiently high score is recorded. New centroids are then computed for each altered cluster. The process is then repeated; at each step of the iteration all the documents are correlated with all the clusters, and the clusters are updated until further updating does not alter any of the existing clusters.

It can be shown that Rocchio's algorithm requires order N^2 vector comparisons, where N is the number of documents, while Doyle's algorithm is of the order N·m where m is the number of clusters. A more efficient method due to Dattola [4] requires time proportional to $N \cdot p \cdot \log_p m$ where N is the number of items in the collection, m is the number of clusters desired and p is the number of clusters produced at each level of the algorithm. The method is an outgrowth of Doyle's attempt to obtain a fast algorithm for clustering large document collections. In each cycle of Dattola's algorithm, each document in the collection is scored against each existing cluster by a certain scoring function. New clusters are then computed while some documents remain loose. The cycle is then repeated with the new clusters. The algorithm is designed to control the number of clusters, size of clusters and amount of overlap. The number of clusters and amount of overlap are specified as input parameters, while the size of the clusters is controlled internally. One problem with Dattola's algorithm is that some way must be found to designate initial clusters.

An inexpensive one-pass clustering method has been proposed by Rieber and Marathe. [5] In this method the first document automatically becomes the centroid of the first cluster. Subsequent documents are correlated with existing clusters and depending on how the correlation with each cluster compares with the minimum correlation cut-off, the document is either admitted to one or more existing clusters or allowed to start a new cluster. If a document is admitted to an existing cluster, the cluster centroid is recomputed. The method allows for disjoint clusters where a document can only be included in one cluster, or overlapped clusters where a document is included in every cluster with which it has a high correlation. The single pass method of Rieber and Marathe compares favorably with more complex clustering methods in search cost, but there is no control on the cluster size or the number of clusters generated. This results in initial clusters being exceptionally large. Moreover, the process is likely to be order-dependent since the formation of the clusters depends on how the documents are encountered.

3. Strategy

To produce retrieval results for the user, two major costs are incurred: the cost of preparing the collection, and the cost of searching. Search cost can be reduced if an investment is made in clustering the collection. The aim of this study is to give a clustering algorithm which operates substantially more cheaply than those presently in use but for which the search costs are similar. In this way it is possible to compare clustering cost directly with search effectiveness. Alternatively, search parameters can be adjusted until search effectiveness is comparable, yielding a direct comparison of clustering and search costs. In either case, it may be possible to exhibit the extent to which the clusters are less optimal than those of other algorithms.

One approach to keeping search costs low is to use an algorithm which, within the constraint of a single pass over the document collection, will produce on the basis of document vector similarities a set of clusters of a given size distribution measured in terms of mean size, maximum size and overlap. This aim is achieved by the algorithm described in this study. The extent to which sets of clusters with similar size distributions have similar search costs is discussed in Section 6. C.

Specifically, the experiments are designed to allow a comparison with Dattola's algorithm. In his work [4], a mean cluster size is chosen. Clusters more than twice or less than half the mean size are not allowed. In the one-pass algorithm described in this study, the mean and maximum are easily controlled. The minimum is not controlled directly, since doing so requires blending small clusters in a second pass. It is questionable whether fixed limits on the cluster size are desirable. Certainly some sort of upper limit is needed to keep search costs well below that for a full search. The effect of a lower limit, given a mean and maximum, is less clear. In any event the method does not control the minimum whereas Dattola's does.

The composition of clusters from a single-pass method depends on the order in which the documents are processed. A document can not be placed in a cluster with a sequence number higher than that of the document. For example, if there are N documents in the collection and n clusters are produced, the first n-1 documents cannot belong to cluster n . In general, it will also be more difficult for a document N to join cluster 1 than for a document with an earlier sequence number.

The effect of order can be controlled partially and indirectly by controlling the rate at which clusters are allowed to form in the early part of the pass. Otherwise, order dependency is inherent in the single-pass method just as it is in other methods inwhich the number of iterations is limited. The degree of order dependency of the algorithm of this study is discussed in Section 6. B.

4. The Algorithm

The clustering algorithm accepts input vectors, each describing a document, and assigns each document either to one or more existing clusters or to a new cluster, depending on the correlation of the input vector with the cluster centroids. Each document vector is of the standard form, consisting of pairs of concept numbers and corresponding weights. The weight of each term in a centroid is obtained by summing the weights of that term for all documents in the centroid.

Vector correlations are computed as the multidimensional cosine between them, COS, as follows

$$\mathrm{COS} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{j=1}^{n} (u_j)^2 \sum_{j=1}^{n} (v_j)^2}} ,$$

where

COS = the cosine correlation between vectors u and v

n = number of terms in the collection dictionary.

COS ranges from 0 to 1. In order to control the cluster size, COS is modified, as discussed below. It is the modified correlation, COR, which is actually used in making clustering decisions.

One stage of the clustering process is defined as the comparison of one input document with all existing clusters. If the correlation, COR, of the document with the centroid of a cluster is greater than the cut-off value, CORCUT, the document is added to the cluster with which it has the best correlation. The document is also added to any clusters for which COR lies no more than an amount GAP below the highest correlation (maximum over all clusters) COR, thus producing overlap. If GAP = 0, no overlap occurs. If a document is not admitted to any cluster, it defines a new one. A centroid is recomputed whenever a document is added.

Number of clusters, cluster size and overlap are controlled dynamically. When, during the course of clustering, the number of clusters compared to the number of documents already processed becomes large, it is necessary to make it easier in general for documents to join clusters. However, to control cluster size it must be generally more difficult for a document to join a large cluster than a small one. To achieve these ends, CORCUT is varied to control the number of clusters while individual correlations, COS, are reduced by an amount related to cluster size in order to control cluster size near some maximum value.

A) Cluster Size

It is desired to define a function COR which will depend on the cosine correlation, COS, and also on the number of documents which the cluster would contain if the new document were admitted. COR should increase with COS and decrease as cluster size increases. If COS is very high, however, it would be unreasonable to exclude the document even from a large cluster. Therefore, when COS is 1, COR should equal 1 as well.

The following function, chosen for the algorithm, meets the above requirements:

$$\text{COR} = \text{COS}^{y}$$

where

$$y = \text{NCEIL}/\ (\text{NCEIL} - \min\ (\text{NCEIL} - 1/B,\ M_i + M_c)+A)/A$$

NCEIL = cluster size ceiling requested by user

M_i = number of documents in input vector
(M_i = 1 unless clusters are being clustered)

M_c = number of documents in cluster

A,B = tuning parameters which the algorithm supplied and the user in general need not be concerned with.

Parameter A controls the rate at which the ratio COS/COR grows with cluster size when a cluster is small. Parameter B controls this ratio near and beyond the cluster size limit. If B is small, clusters can actually grow over the limit given by the user.

B) Number of Clusters

Following each stage, CORCUT is recomputed in order to control the number of clusters finally produced. The ratio of clusters produced to documents processed up to the moment is computed. If this ratio is larger than the value desired in the final result, CORCUT is adjusted downward from a base value FCOR, reducing the probability that a new cluster will be generated in the next stage. If the ratio is smaller than the desired value, CORCUT is raised. The base value FCOR moves toward CORCUT at a rate fixed by the algorithm. The user may supply a value for this rate.

New values of CORCUT and FCOR are computed as follows:

$$\mathrm{CORCUT}_i = (\mathrm{FCOR}_{i-1})^x$$

$$\mathrm{FCOR}_i = \mathrm{FCOR}_{i-1} * R + \mathrm{CORCUT}_i * (1 - R)$$

where

x = (NCL - NCLREQ * NINPUT / NE + D) / D

NCL = number of clusters following stage i-1

NCLREQ = number of clusters requested by user

NE = number of documents in source collection

NINPUT = number of documents input through stage i-1

R = parameter controlling rate at which FCOR follows CORCUT

D = tuning parameter set by algorithm.

If R=0, CORCUT varies widely and poorer clusters are produced.

C) Overlap

A document is added to a cluster only when two conditions are met:

1. COR > CORCUT
2. max(over all clusters) COR - COR < GAP

It can be seen that GAP controls overlap. Between stages GAP is adjusted as follows:

$$GAP_i = (FGAP_{i-1})^z$$

where

$$z = (M_1 + \ldots + M_{NCL} + E)/(NINPUT*(OVLAP + 1) + E)$$

$OVLAP$ = the value requested for

$$\left(\sum_{i=1}^{NCL_{NE}} M_i/NE\right) -1$$

$FGAP$ = base value of GAP

$$FGAP_i = FGAP_{i-1}*R + GAP_i*(1 - R)$$

D) An Example

The following example illustrates how the clustering parameters are adjusted dynamically during the clustering process and how the individual correlations are computed. The values presented are taken from a run on the ADI collection. User selected parameters are given as follows:

NCEIL = 15

NCLREQ = 9

OVLAP = .122

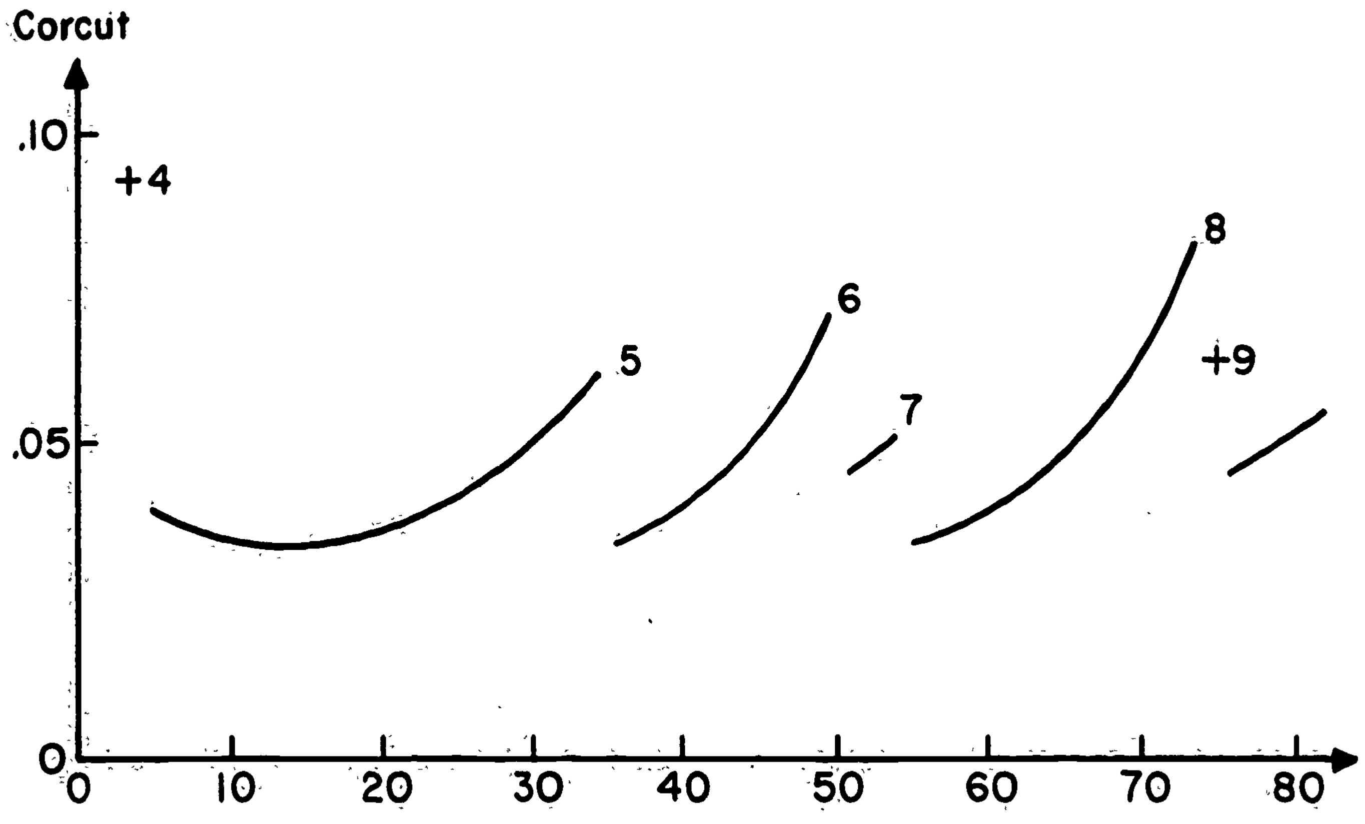
Default options are used for the other parameters, namely:

FCOR = .4	A = 40
FGAP = .001	B = 2
R = .9	D = 5
	E = 1

Fig. 1 shows how CORCUT varies during clustering. Fig. 2 gives a similar presentation for GAP. Of course, CORCUT and GAP change in discrete steps after each document has been clustered. Points in the figures have been connected for ease of presentation.

Consider, for example, the input of the 65th document. Seven clusters exist at this point, so COS and COR are computed with respect to each cluster, giving the following results:

Cluster Number	Number of Documents in Cluster	COS	y	COR
1	14	.458	2.5	.142
2	10	.080	1.3	.037
3	14	.498	2.5	.175
4	12	.214	1.5	.010
5	12	.205	1.5	.009
6	1	.232	1.17	.181
7	1	.050	1.17	.030

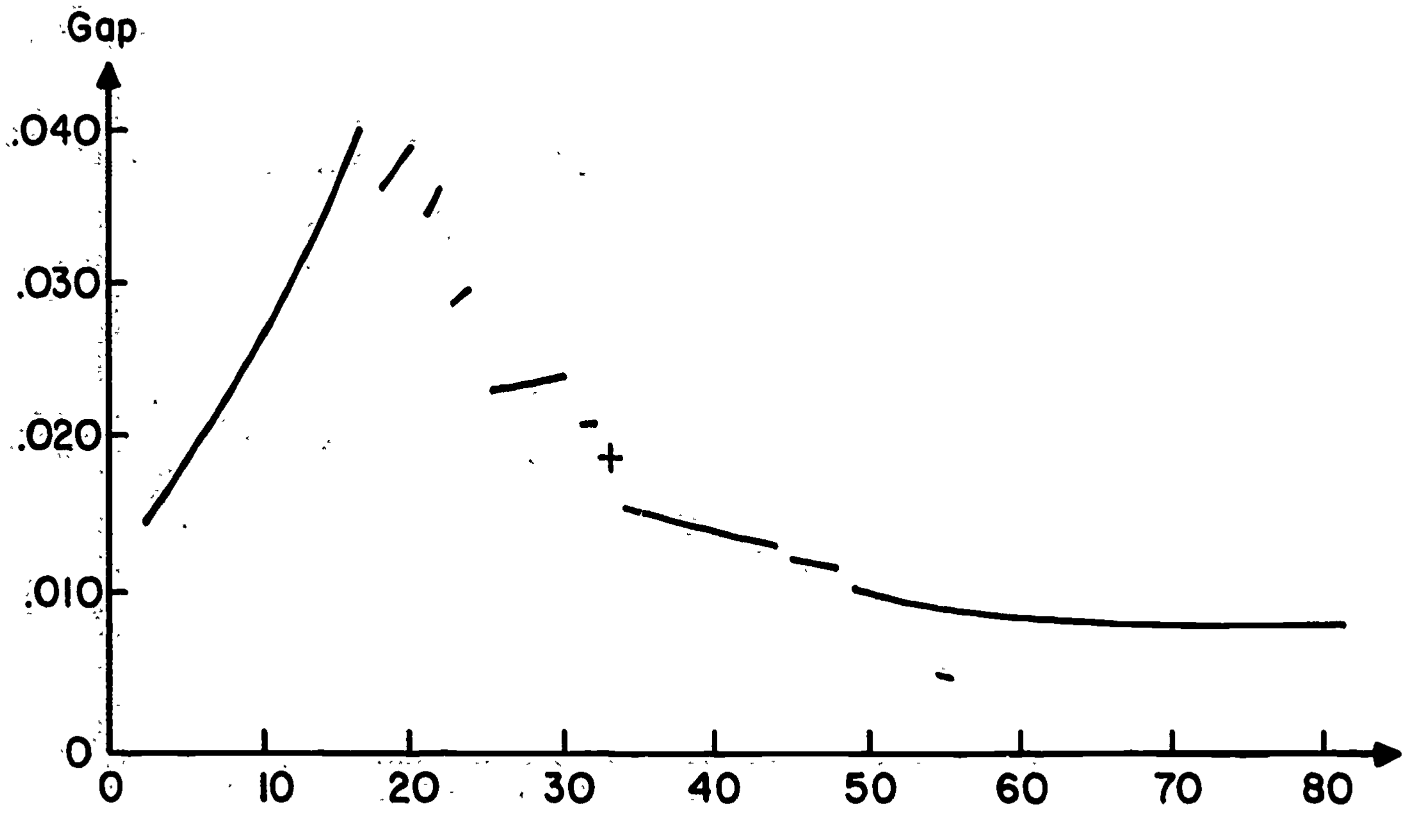


Number of Documents Input

Variation of Corcut with Input

Points at which new clusters form are marked with cluster numbers. First three points are off scale.

Fig. 1



Number of Documents Input

Variation of Gap with Input

Each downward discontinuity in the plot occurs when a document is assigned to multiple clusters

Fig. 2

Taking the values of CORCUT = .045 and GAP = .0089 as adjusted following the 64th document, document 65 is admitted to clusters 3 and 6 as follows:

Cluster Number	Cor	Admit	
6	.181	*	
3	.175	*	
1	.142		<— .181-Gap =.172
2	.037		<— CORCUT = .046
7	.030		
4	.010		
5	.009		

5. Implementation

The algorithm is implemented in Fortran. The user specifies starting values as follows:

NCEIL = approximate maximum number of documents desired in any cluster

NCLREQ = number of clusters desired

OVLAP = fractional overlap desired

The algorithm sets the following parameters:

FCOR = .4	A = 40
FGAP = .001	B = 2
R = .9	D = 5
	E = 1

The user may override these values if he desires. It is expected that extensive use of the algorithm would lead to better values than those already found, although the algorithm is not highly sensitive to them.

A) Storage Management

The algorithm is designed with a veiw toward application to large collections. Core storage and accesses to secondary storage are minimized in the following ways. Clusters are stored in sequential locations rather than in a 2-dimensional array. Only sufficient core storage for the clusters as a group need be alloted regardless of the variation in cluster size. A linked list could also be used. However, if secondary storage has to be used to store part of the cluster information, sequential storage is them preferable.

Sequential storage requires moving the cluster information in order to insert new concept-weight pairs. To minimize moves, two consecutive input vectors are kept in core. In the course of one stage of the algorithm, the input to the previous stage is added to the appropriate clusters and correlation of the current document is made simultaneously. The entire cluster collection is moved at most once for each stage.

6. Results

This study employs two document collections to evaluate the performance of the single-pass algorithm, the ADI collection of 82 documents and 35 queries and the Cranfield collection of 424 documents and 155 queries. Evaluation is made by comparing the results using the present algorithm with the results using Dattola's algorithm. Several clustering and search runs*

*In the recall-presicion curves presented, the modifications proposed by Dattola, [4] pp. 16-24, to compensate for variations in correlation percentage and uniform distribution of unrecovered relevant documents are incorporated.

were made using similar input parameters in both algorithms. Full search results on the ADI collection are also shown for comparison.

The cost of clustering is first discussed in Section 6. A. Several clustering runs were made using the single-pass algorithm over different orderings of the ADI collection to show the effects of the order in which documents are processed. This is discussed in Section 6. B.

Clustering is also done at two levels as well as one level to illustrate savings in clustering time and to show the effect of multilevel clustering on cluster quality. The results of searching the ADI collection, including multilevel search, are discussed in Section 6. C. Finally, the algorithm is applied to the Cranfield 424 collection and the results are discussed in Section 6. D.

A) Clustering Costs

Cost comparisons between clustering methods can be made according to several criteria. The two for which results are presented in this study are:

1. Number of vector comparisons performed.
2. Computer resources used, mainly CPU and I/O time.

The first criterion allows comparison of algorithms to a large degree independently of the programming techniques employed and the system in which the programs are embedded. The second reflects the effects of the system and the programming techniques as well.

Consider the number of vector comparisons. It is convenient to assume that clusters are formed at uniform intervals during the processing of the collection, that is, cluster 1 forms with document 1 and in general cluster i forms with document $\frac{N}{m}(i-1) + 1$ and so forth, where there are N documents and m clusters produced. Under this assumption, the number of comparisons for clustering on a single level is given as follows:

$$C_1 = \frac{N - 1}{m} \sum_{i=1}^{m} i = \frac{(N - 1)(m + 1)}{2}$$

where

C_1 = number of comparisons on level 1

N = number of documents in the collection

m = number of clusters produced

If clustering is done on x levels and p clusters are formed at each level from each cluster on the next higher level, the number of comparisons made at level i is

$$C_i = \frac{(N - 1)(p + 1)}{2} \quad .$$

Consequently, for multilevel clustering over x levels, the total number of comparisons C is

$$C = \sum_{i=1}^{x} C_i = \frac{(N - 1)(p + 1)}{2} x$$

and since $m = p^x$,

$$C = \frac{(N - 1)(p + 1)}{2} \log_p m \quad .$$

Dattola gives a similar derivation in complete detail [4], giving the total number of comparisons D over x levels as defined above:

$$D = kNp \log_p m$$

where

k = the average number of times a set of documents is compared to a set of cluster centroids.

A typical value of k is 16 for a collection the size of the Cranfield 424.

Based on vector comparisons, then, one could expect an increase in speed over Dattola's algorithm of 2k for large p and $\frac{3}{2}k$ for $p = 3$ (the optimum value proposed by Dattola [4]). A somewhat lower ratio is shown in the results of this study. Table 1 shows comparative results. For the ADI collection the ratio of number of comparisons is 15:1, implying a k for Dattola's algorithm of 7.5. For the Cranfield 424 the values are 8.2:1 and 5.25, respectively. The difference from the value $k = 16$, which is expected, is largely explained by the following factors:

1. In the runs using the present algorithm, a burst of clusters was forced to form at the outset. Thus the uniform formation assumption is not met and more comparisons are made with the present algorithm than predicted. In the limiting case where the first m documents form the m clusters, C is bounded as follows:

$$C < (N - 1)p \log_p m \quad .$$

2. During an iteration of Dattola's algorithm the number of trial centroids is frequently less than the chosen value of p .

ADI Collection (82 documents)	Present Study	Dattola's Algorithm
Number of clustering runs over which the following results are averaged	11	1
Vector comparisons	445	6614
CPU seconds (360/65)	5.8	44.0
I/0 seconds (360/65)	13.3	38.0

Cranfield 424 Collection* (424 documents)	Present Study	Dattola's Algorithm
Number of clustering runs over which the following results are averaged	1	1
Vector comparisons	5579	45840
CPU seconds (360/65)	214	439
I/0 seconds (360/65)	126	653

Clustering Cost Comparisons Between the Present Study and Dattola's Algorithm

Table 1

*Results shown for Dattola are for two-level clustering

3. Iterations of Dattola's algorithm sometimes converge before the iteration limit is reached.

4. Two levels of clustering were performed on the Cranfield 424 under Dattola's algorithm against a single level for the algorithm of this study.

The comparison of CPU times in Table 1 is somewhat less favorable to the algorithm of this study, particularly on the Cranfield 424 collection. The major factor is scoring, or vector comparison. Scoring may be done over an array of weights if concept numbers are restricted to a prescribed range. In the case of Dattola's implementation, numbers do not exceed 10,000 so comparison may be done over a 10,000-element array in which the concept number is given by position rather than in a list where the concept numbers appear explicitly. The present algorithm uses such a list, ordered by concept number, and runs more slowly as a consequence.

Perhaps the most important point to be made in this discussion is that the algorithms of Dattola and of the present study are of the same order. The constant multiplier k , however, is of the order 16 in the case of Dattola and 1 in the case of the present study.

B) Effect of Document Ordering

The effect of ordering is studied by comparing the search results on the original ADI collection and three permutations of it. The three permutations are constructed by reordering the collection according to tables of random numbers between 0 and 99.

Clusters are generated using NCEIL = 22, NCLREQ = 9 and OVLAP = .122 (default options are used for the remaining parameters). A minimum of 10 documents is searched for each query. Plots of precision vs. recall are shown in Figures 3 and 4. As expected, there is significant variation in the performance of the algorithm for different orderings of the documents. It is interesting to note that the original ordering of the ADI collection gives the worst results. The third permutation gives the best results which actually exceed Dattola's results except in the high recall region. It can be seen that the relative improvement of the results by reordering is better in the document-level average plots than in the recall-level average plots.

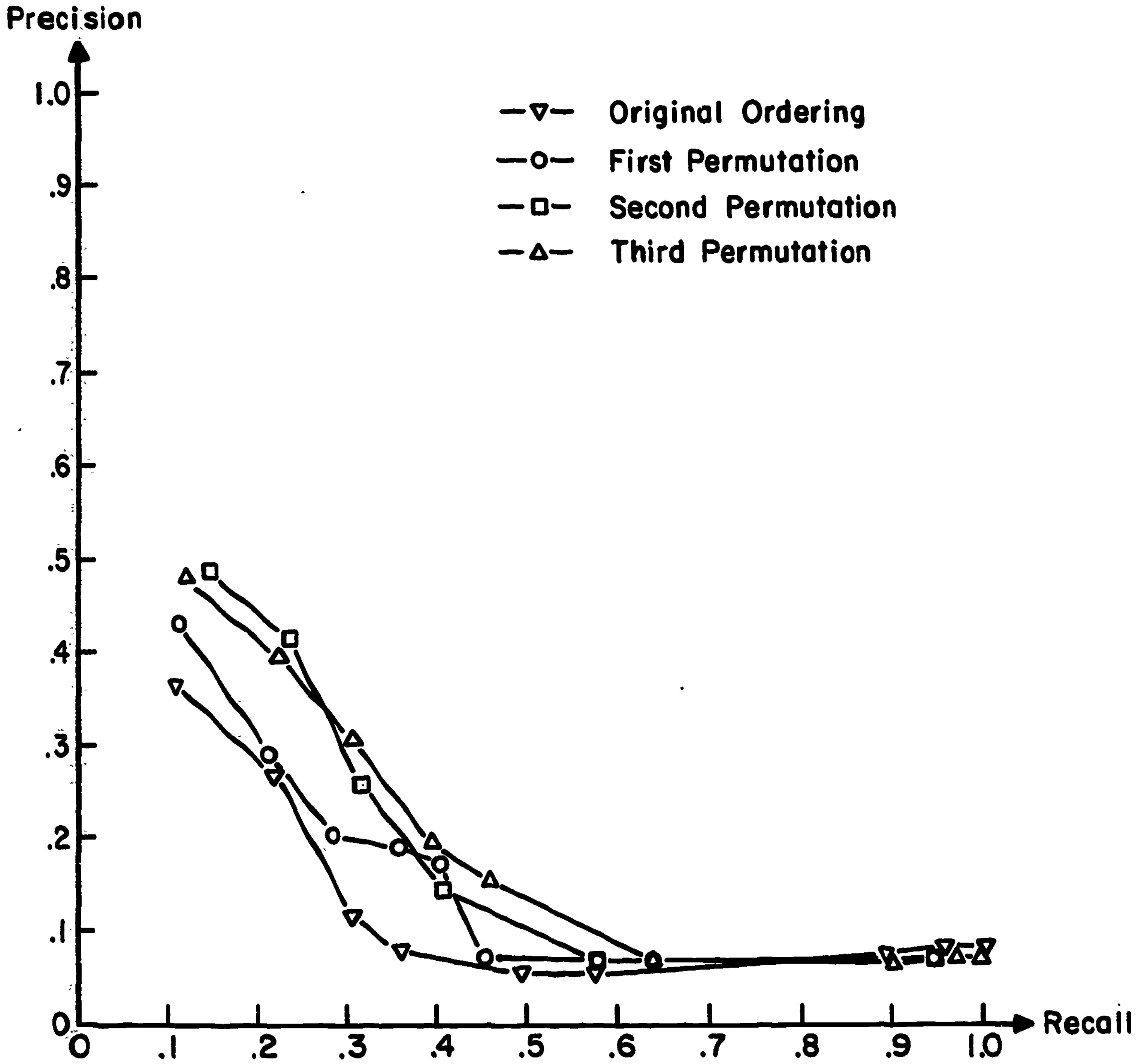
Figures 5 and 6 show the spread between the curves representing the original ordering and the third permutation of the documents of the ADI collection. The results can be compared with those of a full search and clustered search using Dattola's algorithm.

Multilevel clustering is discussed in Section 6. C. Since multilevel clustering improves search results, it is possible that document reordering may be used to obtain further improvement in this case, although it would be difficult to determine a suitable ordering in advance.

C) Search Results on Clustered ADI Collection

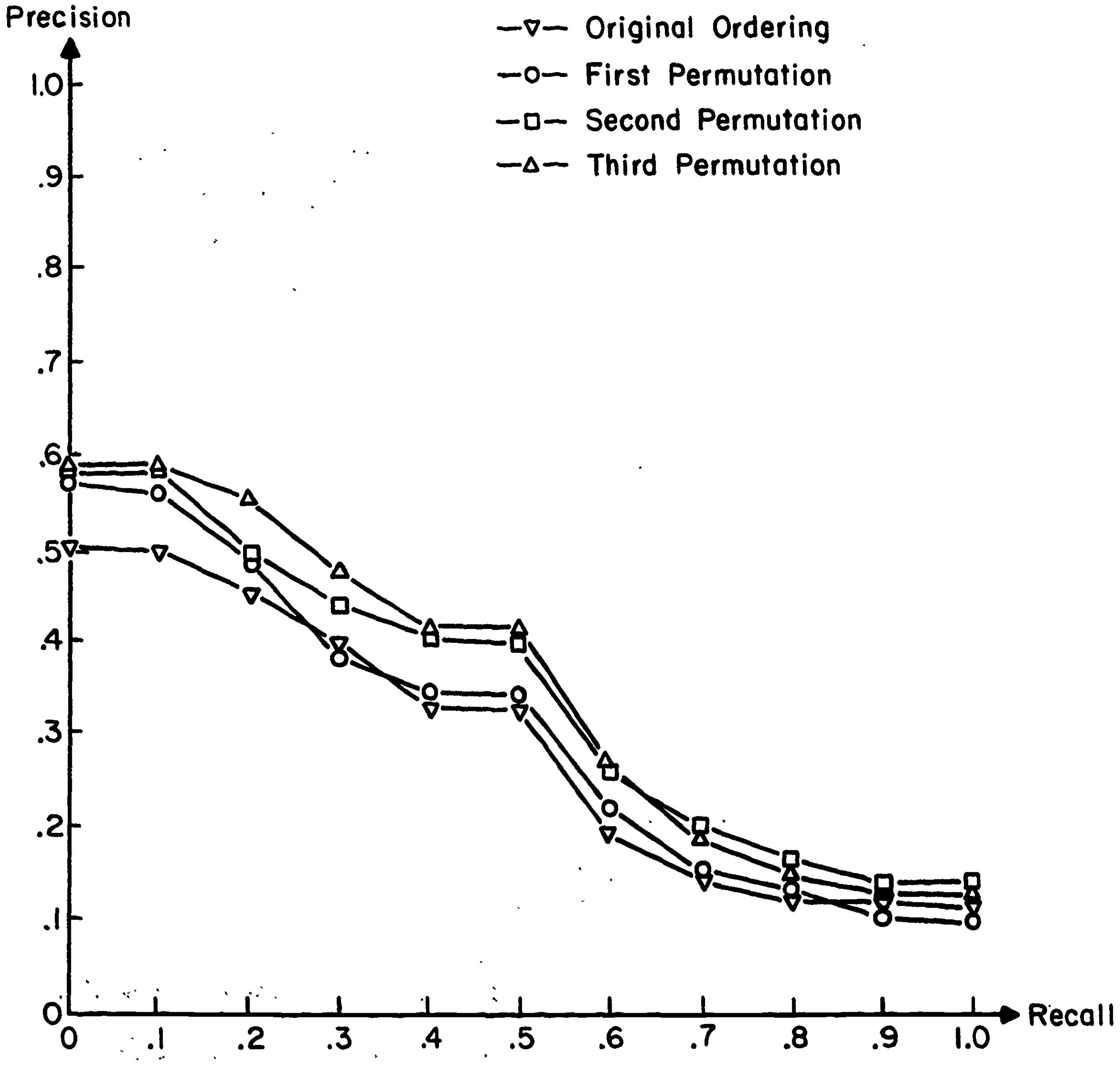
Recall and precision plots of searches on the clustered ADI collection are shown in Figures 5 and 6. As discussed in Section 6. B, it may be seen that there is a substantial variation in the quality of the clusters over different orderings of the collection, as measured by search results. In comparison with both the full search and Dattola's algorithm, the present algorithm shows a tendency to perform best in the low-recall region. This effect may be observed in all results of this study and, consequently, it is a distinguishing characteristic of the single pass algorithm.

It should be observed that search costs for the results shown using



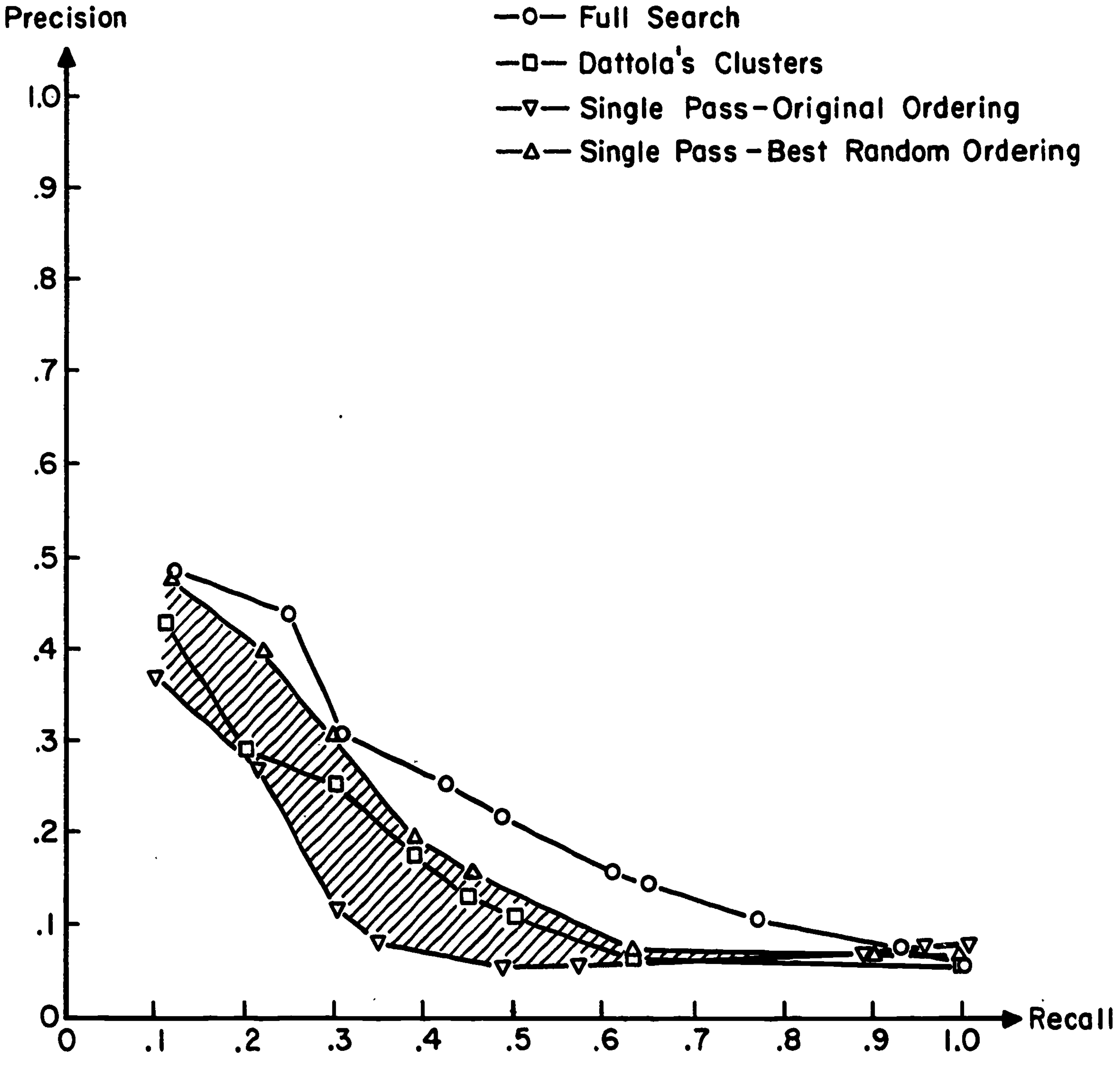
Effect of Ordering on Document Level Averages in Search of ADI Collection Clustered with Single Pass Algorithm

Fig. 3



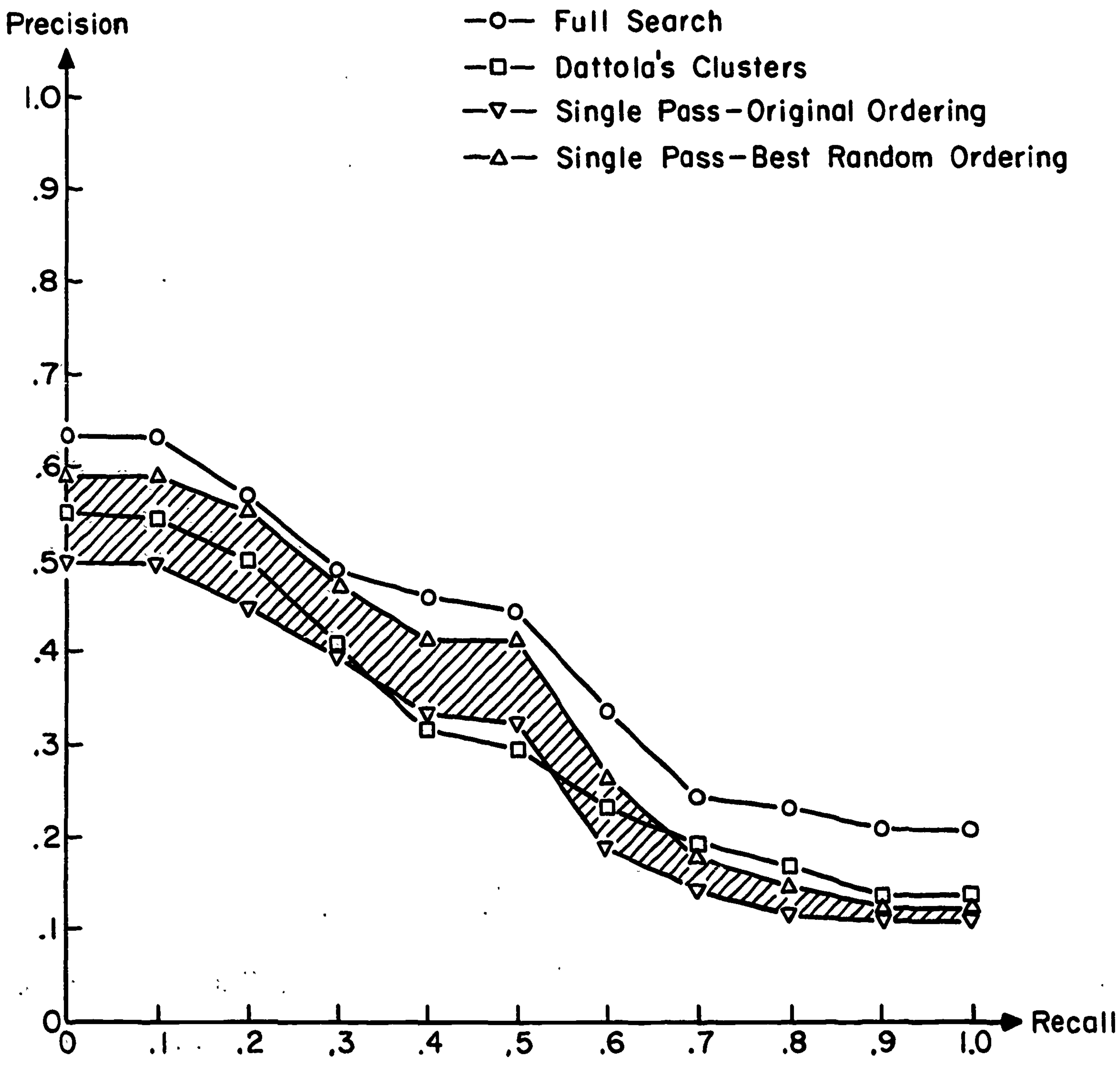
Effect of Ordering on Recall Level Averages in Search of ADI Collection Clustered with Single Pass Algorithm

Fig. 4



Document Level Averages for Full Search, Dattola and Single-Pass Clustered Search on ADI Collection

Fig.5



Recall Level Averages for Full Search, Dattola and Single-Pass Clustered Search on ADI Collection

Fig. 6

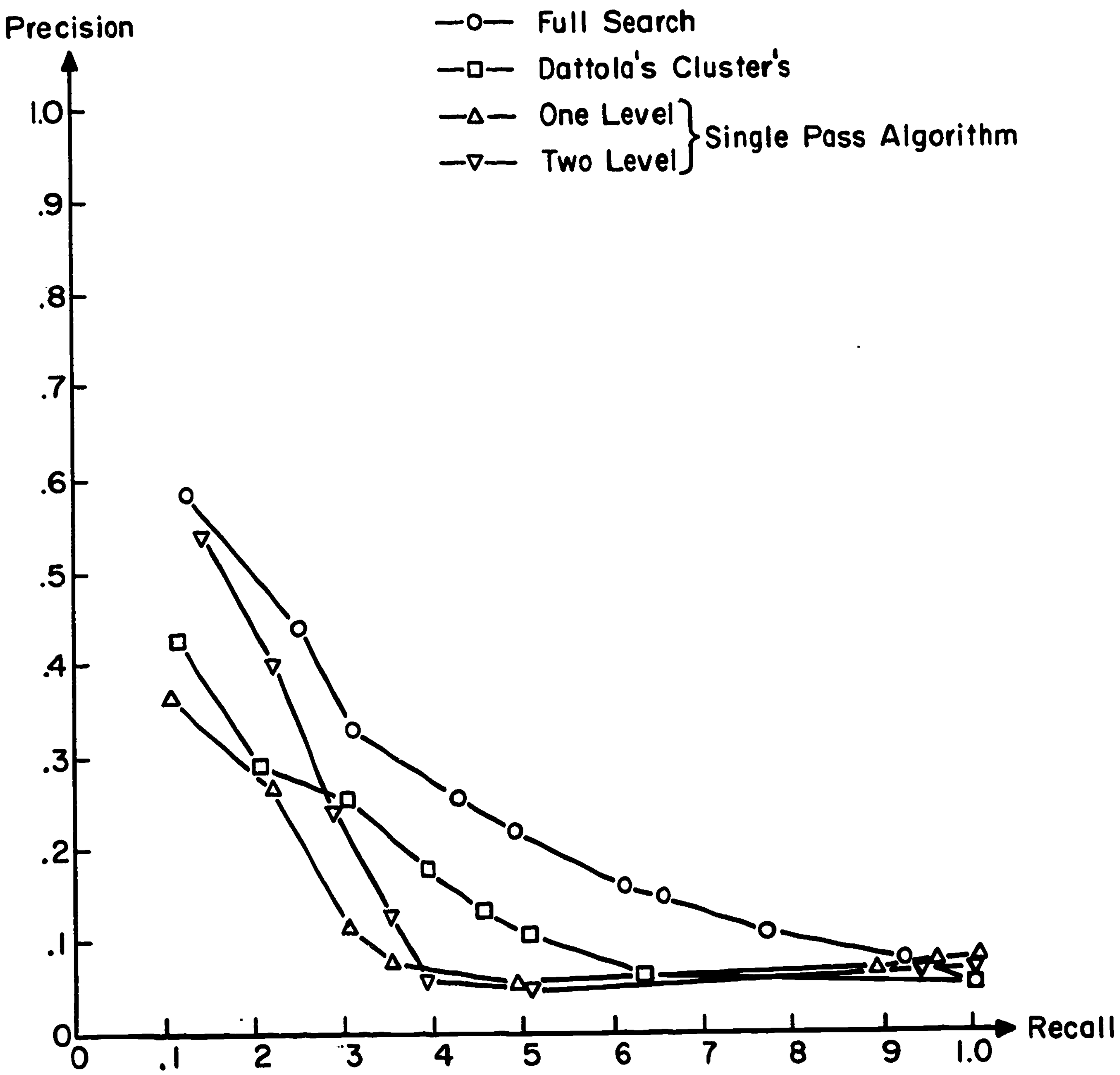
the present algorithm and the results using Dattola's clusters are roughly similar. For example, 990 vector comparisons are made in searching the clusters produced from the ADI collection as originally ordered compared with 966 vector comparisons in searching Dattola's clusters.

In Section 6. A an expression for the cost of clustering is given. The relationship is such that clustering cost is reduced both for the algorithm of this study and for Dattola's algorithm if multilevel clustering is done.* It is not known how the search results on clusters formed at a single level and over multiple levels compare in the case of Dattola's algorithm. However, in the case of the present algorithm, multilevel clustering is not only less expensive to perform but it can produce markedly better clusters.

Figures 7 and 8 show the improvement in recall and precision achieved when the same ordering of the ADI collection is clustered over one level and two levels. Clusters are formed at the first level and then sons of each are formed at the second level. The ordering chosen was the one among the four tested for which recall and precision are poorest for the single level clusters. It is suspected that improvement would also be shown for the other orderings.

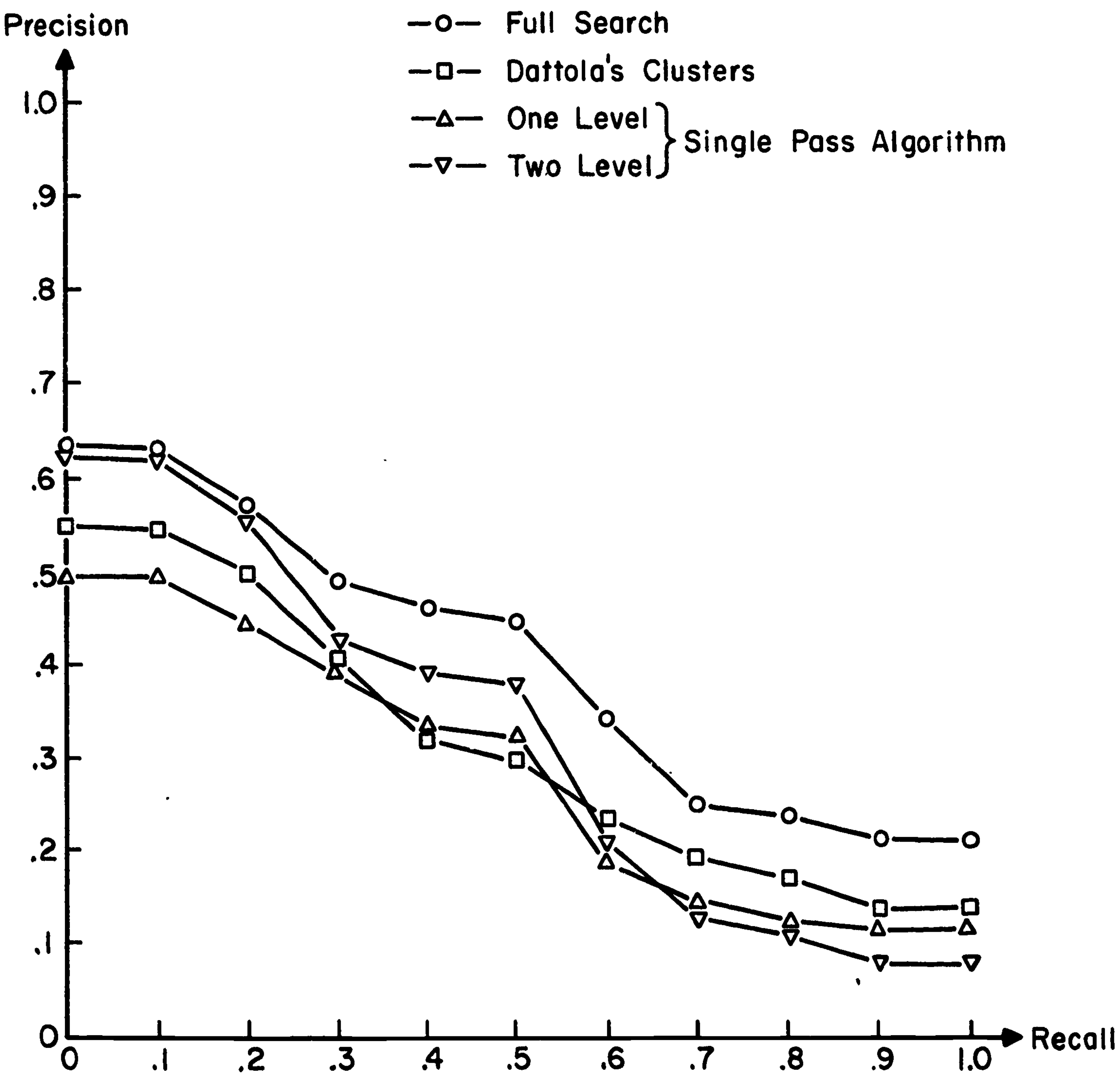
It can be noted in Figures 7 and 8 that the two-level clustered search is markedly better than the one-level search, particularly in the low recall region. Moreover, the two-level search performs better than the search on Dattola's clusters in the low recall region and approaches the full-search curve in this region.

*It is thought proper to apply the description "single pass algorithm" to multilevel clustering where (a) each level is clustered in a single pass and (b) the multilevel algorithm performs fewer comparisons than a single pass would perform as a single level.



Document Level Averages for Full Search, Dattola, One-Level and Two-Level Clustered Search of ADI Collection

Fig.7



Recall Level Averages for Full Search Dattola, One Level and Two Level Clustered Search of ADI Collection

Fig. 8

In general, the probability that a document late in the input enters an early cluster is related to the number of clusters formed in the pass over a level of the cluster tree. The greater the number of clusters, the more the membership of early clusters is confined to documents early in the collection. This can be seen by observing that, except for the case where a collection is partitioned into m sequential clusters, a cluster on the average allows documents to be admitted over a sequence of input documents larger than the cluster size, that is, for a single level of clustering,

$$n = a \frac{N}{m}$$

where

n = number of documents for which a substantial probability of admission to a certain cluster exists.

a = a constant (a > 1).

If clustering is done over two levels,

$$n_j^{(2)} = a \frac{n^{(1)}}{p}$$

where $n_j^{(2)}$ is the jth son of $n^{(1)}$ (superscripts refer to the level of the tree). Thus, at the final level x there are m clusters and

$$n^{(x)} = a^x \frac{N}{p^x} = a^x \frac{N}{m}$$

$$n_j^{(x)} > n_j^{(1)} \quad \text{for} \quad x > 1 \quad \text{and} \quad a > 1$$

The above suggests that producing as few as two clusters from each father in the cluster tree would allow the best associations to be made. It is also suspected that better results may be obtained if the order in which the documents are compared at each level is reversed. The rationale here is that the last documents admitted to the cluster have in general the highest correlation with the centroid. It is hypothesized that in a pass in reverse order, these documents will tend to form a single cluster and allow the earlier ones to fall in separate groupings.

The search results reported above suggest that sets of clusters with similar size statistics have similar search costs when the same search parameters are used. Comparative figures are shown in Table 2.

Overlap figures are also given in Table 2. It may be seen that overlap varies substantially between the sets of clusters compared. Certainly, increased overlap increases search costs since it increases average cluster size. Whether increased overlap affects recall and precision curves to any great extent is less clear. It may be argued that the clustering algorithm operates without knowledge of the query set subsequently used to search the collection and, consequently, assignment of documents to multiple clusters is independent of relevance judgments. Unpublished results of Dattola in which overlap was varied widely when clustering both the ADI and Cranfield 200 collections without apparent effect on recall and precision support the hypothesis of independence.

The results of this study as well suggest that overlap is uncorrelated with recall and precision. Overlap is not held constant in the results presented because of the difficulty in matching the overlap measures of Dattola's algorithm and of the algorithm of this study. As may be seen

	ADI Collection (as ordered)			Cranfield 424	
	Single Pass		Dattola	Single Pass	Dattola 2-level
	1-level	2-level	1-level		
Fractional overlap	.17	.33	.01	.14	.23
Average cluster size (No. of documents)	10.7	12.1	9.2	24.2	23.7
Average cluster size/ collection size	.13	.148	.112	.057	.056
No. of clusters	9	9	9	20	22
No. of clusters/ collection size	.11	.11	.11	.047	.052
No. of vector comparisons (Search Cost)	990	1105	966	13,103	12,200
Fractional search cost	.35	.39	.34	.20	.186
No. of documents sought in search	10	10	10	43	43

Summary of Cluster Statistics
and Search Costs

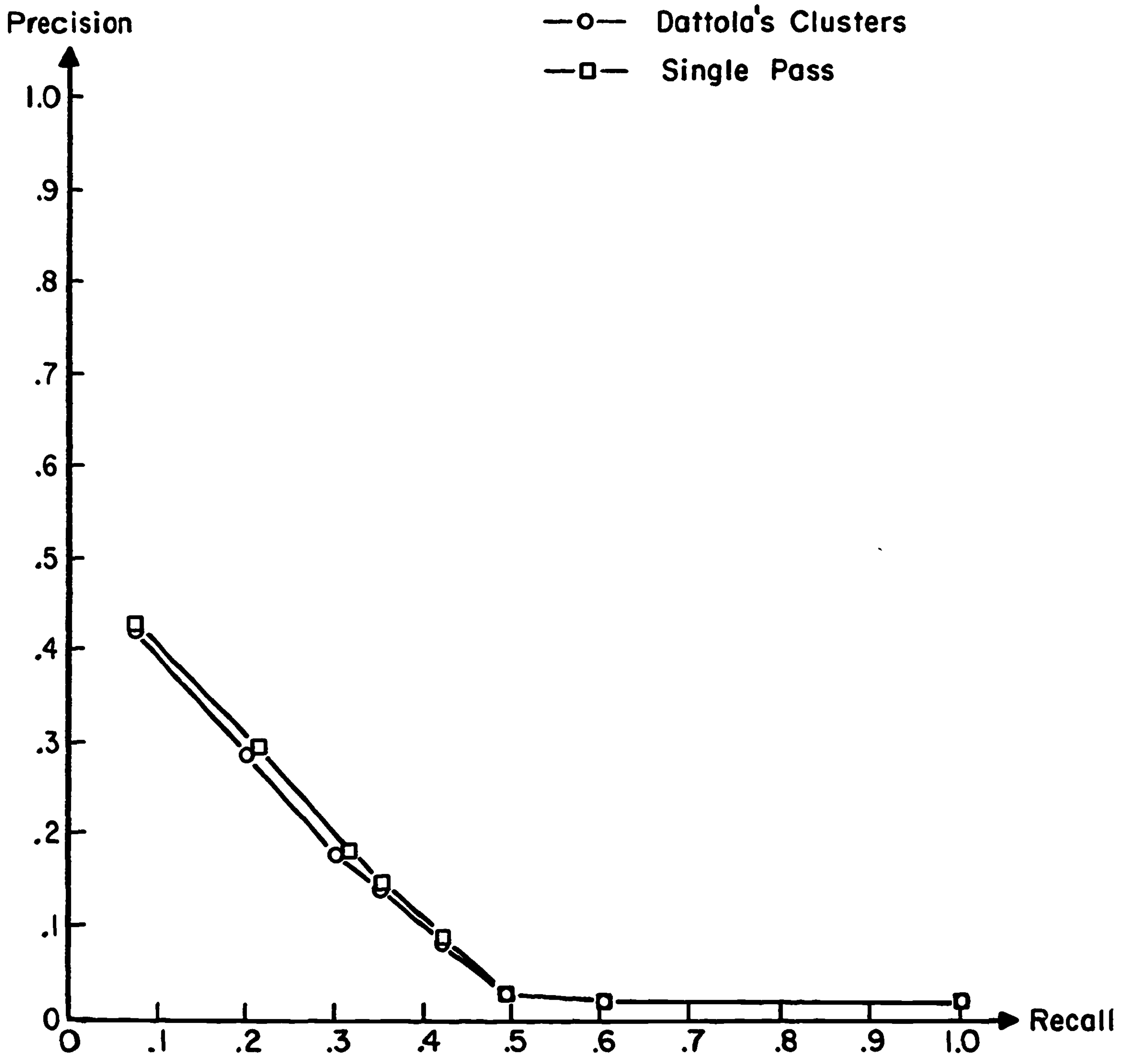
Table 2

in Table 2, higher overlap values occurred with the better results on the ADI collection, but the reverse is true for the two runs made on the Cranfield 424 collection.

D) Search Results of Clustered Cranfield Collection

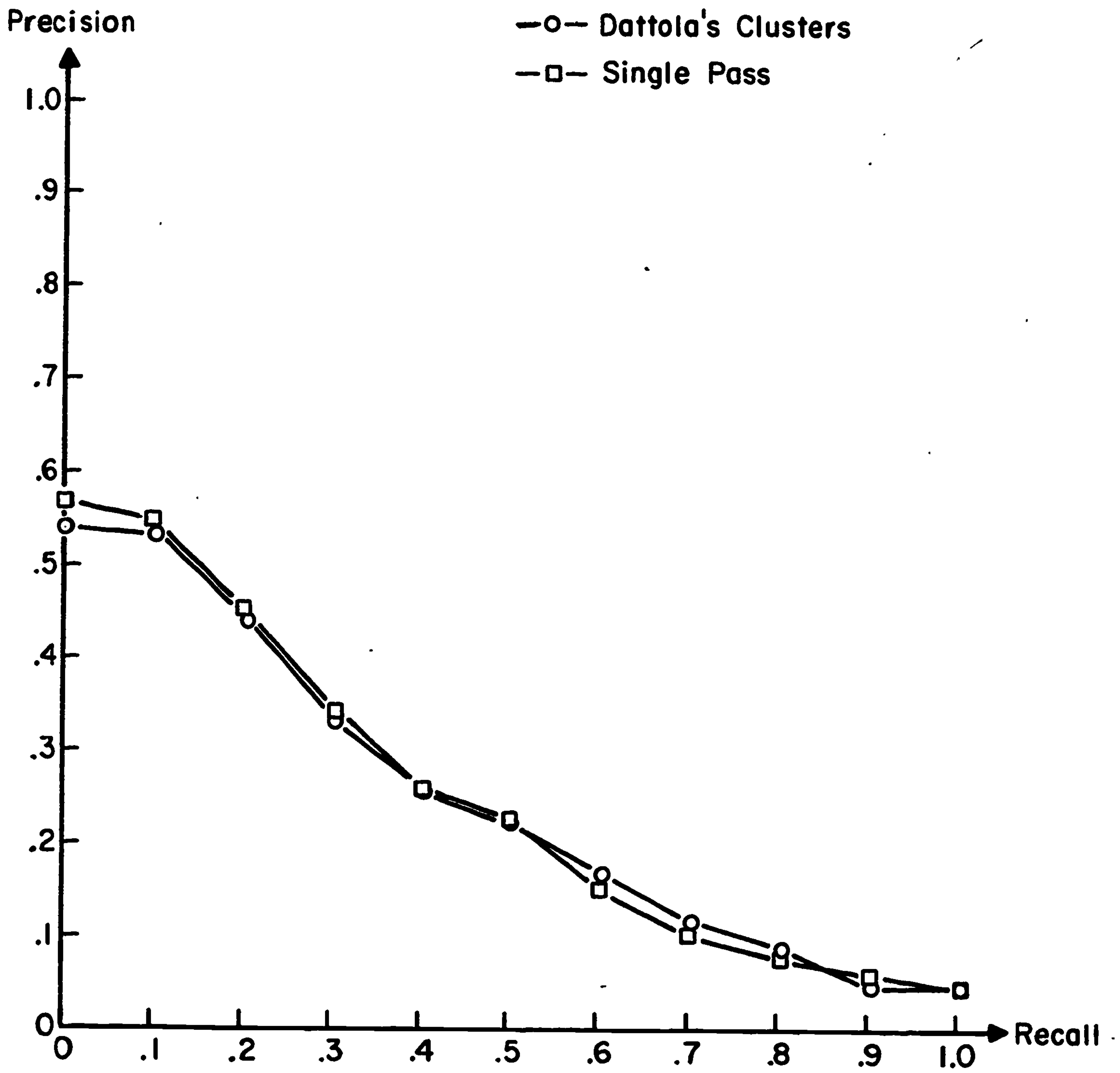
The Cranfield 424 collection is known to have sequential groupings of some of the documents relevant to certain queries. Consequently, a random ordering of the collection is used as input to the single level clustering run performed with the algorithm of this study. Figures 9 and 10 show the recall and precision curves from a search on this set of clusters compared with a search on a set of clusters produced with Dattola's algorithm. In the case of Dattola's algorithm [4] clustering is done over two levels using clustering parameters which were found to be optimal on the ADI and Cranfield 200 collections.

As may be seen from Figures 9 and 10 the algorithm of this study produced a slightly better recall and precision curve than Dattola's. Additional runs on several permutations of the collection are needed to establish whether a significant difference is shown consistently. Search costs on the Cranfield 424 collection using the single pass algorithm are comparable to search costs using Dattola's clusters. As shown in Table 2, the present algorithm requires 13,103 vector comparisons, compared with 12,200 for Dattola's case. A more useful measure, which allows comparison of collections of different size, is the fractional search cost, defined as the number of vector comparisons per document per query. Fractional search costs on the Cranfield collections for the single pass method and for Dattola's are roughly similar, as seen in Table 2.



Document Level Averages for Dattola and Single Pass Clustered Search of Cranfield 424 Collection

Fig. 9



Recall Level Averages for Dattola and Single Pass Clustered Search of Cranfield 424 Collection

Fig. 10

7. Conclusions

The single pass algorithm of this study is substantially less costly to execute than other clustering algorithms, Dattola's in particular. As may be expected, however, the quality of the cluster set depends to a large degree on the order in which the documents are encountered by the algorithm. On the ADI collection, some orderings produce search results, measured by recall and precision curves, better than Dattola's clusters in the low recall range. Other orderings are worse at virtually all points on the curve.

A single clustering run is reported for a larger collection, the Cranfield 424. It indicates that cluster quality is not degraded when the algorithm is applied to a collection about five times as large as the ADI.

Multilevel clustering using the basic single pass approach of this study is shown both to be cheaper and to produce substantially better clusters as measured by search results. There is a strong suggestion that further work could establish the multilevel single pass algorithm to be as good or better than Dattola's algorithm for most orderings of the collection.

The basic limitations of the single pass method appear to be overcome best when multilevel clusters form a binary tree. It is possible that the contents of a collection would dictate nodes in the tree of higher degree. The trade offs involved in such cases should be investigated.

The multilevel clustering of this study is confined to presenting only the lowest level of the tree to the search algorithm. However, the entire tree could easily be made available to the search algorithm. If so, two possible ways of construction present themselves. The first is to fetch each document description from the collection only once. Each document would be passed down the tree and all decisions relative to it would be made in sequence, level by level. In effect, many levels of single pass clustering would be carried on simultaneously, each cluster at each level being treated as the collection from which its sons are formed. In the case of a binary tree, for example, the first document in the collection would be passed down the entire leftmost branch of the tree to the predetermined lowest level. It would define, at first, the leftmost cluster at each level. The next document would then be passed down the tree. As long as it was admitted to existing clusters it would add to their definition. At the point (if any) where it was selected to define its own cluster, it would form a right branch and then a sequence of left branches down to the lowest level. Subsequent documents would be processed similarly.

The second method of construction is to form each level completely before the next level is begun. Document descriptions would have to be fetched from the collection once for each level plus additional occurrences caused by overlap. However, cluster quality might well be better since the direction of passing over the documents can be reversed at each level.

Even when just two sons are formed from each father in the tree, there still exists the possibility that natural clusters will be split into fragments. By the nature of the process, once two documents are separated, they cannot be associated again. To a certain degree, searching over multiple clusters allows these documents to be found. It would be best, however, to have them properly associated in the tree. It is proposed, therefore, that the leaves of a completed tree be compared one to another. Those with particularly high correlations would be coalesced into a single cluster taken to be the son of both fathers. Such a coalescing process would deform the tree only at the lowest level and could be expected to reassociate sets of documents which were of roughly equal size and large enough to be a majority of the members of the clusters involved. Any further investigation will necessarily include experiments designed to strengthen, if possible, the results already found and to consider further the selection of optimum clustering parameters.

References

[1] D. Williamson, R. Williamson and M. Lesk, The Cornell Implementation of the SMART System, Information Storage and Retrieval, Report ISR-16 to the National Science Foundation, Section I, Department of Computer Science, Cornell University, September 1969.

[2] J. J. Rocchio, Jr., Document Retrieval Systems: Optimization and Evaluation, Harvard Doctoral Thesis, Information Storage and Retrieval, Report ISR-10 to the National Science Foundation, Harvard Computation Laboratory, Cambridge, March 1966.

[3] L. B. Doyle, Breaking the Cost Barrier in Automatic Classification, SDC paper SP-2516, July 1966.

[4] Dattola, R. T., Experiments With a Fast Algorithm for Automatic Classification, Information Storage and Retrieval, Report ISR-16 to the National Science Foundation, Section XIII, Department of Computer Science, Cornell University, September 1969.

[5] V. P. Marathe and S. L. Rieber, A One-Pass Clustering Algorithm, Information Storage and Retrieval, Report ISR-16 to the National Science Foundation, Section XIV, Department of Computer Science, Cornell University, September 1969.

XIII. A Systematic Study of Query-Clustering Techniques: A Progress Report

S. Worona

Abstract

An experiment using various techniques of query clustering on the Cranfield 424 document collection is described and some preliminary results are given. Several methods of evaluating the performance of clustered searches in the context of query-clustering are discussed. Finally, some observations are made concerning use of the SMART system as implemented at Cornell University.

1. Introduction

The idea of query-clustering as an aid to information retrieval systems is first defined and examined by V. R. Lesser [1]. In that report, a two-level clustering algorithm is described in which members of a document collection are assigned to clusters according to their relationship with previously-formed clusters of queries.

It is argued that there are three advantages to performing the clustering in this manner; first, the accuracy of a given search procedure may be increased by comparing queries to sets of related queries already processed by the retrieval system, instead of sets of related documents. Second, it is likely that such a system will perform better as time passes and more queries are available for clustering. Finally, because the cost of most clustering algorithms increases with the size of the collection being clustered, it is more economical to use a query collection than the associated — and much larger — document collection. A more thorough general discussion of the first two of these points can be found in [2], as well as in the original paper by Lesser. Applications of these ideas to various methods of query clustering are discussed below. (For additional information on clustering algorithms, see [3,4,5,6,7,8]. Background material and further references may be found in [9].)

The general process of query clustering may be divided into three parts, or "phases":

1) generation of query clusters;
2) generation of document clusters from the query clusters of phase 1; and
3) definition and assignment of centroids for the phase 2 document clusters.

Each of these three phases may be accomplished in several different ways. A combination of three such methods — that is, one for each phase — is termed an "implementation" of query clustering, or a particular query-clustering technique.

Many procedures exist for performing phase 1, that is, in the initial clustering job. The variables in using these algorithms include the number of clusters desired, the amount of overlap permitted, and the number of queries to be clustered. The last parameter is particularly important to a query-clustering technique, because it is hoped that search results improve with an increase in the number of queries clustered.

Phase 2 may be implemented in any of the following three ways:

1) Relevancy decisions

2) Correlation with query centroids

3) Correlations with clustered queries.

In the first case, documents which are relevant to one or more queries in any one query cluster are grouped. For case 2, all documents are correlated against each query centroid, and those documents correlating highly with any such centroid are put into one cluster. In the third case, the documents are correlated with all queries used in clustering, and a document cluster is formed from those correlating highly with one or more queries in any given query cluster. In all three methods, one document cluster is formed for each query cluster generated in phase 1. The query cluster from which such a document cluster is formed (whether by relevancy decisions, centroid correlations, or query correlations) is called its _underlying query cluster_.

There is much disagreement about the manner in which a centroid is chosen to represent the documents in a cluster. Concepts may be logical or weighted, with very high or very low weights arbitrarily either retained or dropped by one of several methods [10]. In query clustering, however, the choice of the type of centroids used (phase 3) is much more basic — the centroids for each phase 2 cluster may be either the _document centroid_ formed from the documents of the cluster, or the _query centroid_ of the underlying query-cluster.

There are obviously a large number of query-clustering techniques which may be formed from different combinations of the above variations of the three phases. At this time, the only available studies of query clustering are focused on varying phase 1 methods, while using relevancy decisions for phase 2 and query centroids in phase 3. This paper deals with an experiment which considers all six combinations of second- and third-phase variations, while also using different numbers of queries in phase 1. This produces eighteen different query-clustering techniques, which are then compared to a standard full search, and also to a set of "normally-generated" clusters (formed using Dattola's Algorithm). Because of the large volume of data generated by the experiment, a thorough analysis of the results is not yet available. The present report will be followed by such an analysis, using the parameters developed in [2].

2. The Experiment

A) Splitting the Collection

In all experiments with query clustering, it is first necessary to divide the query collection used into two disjoint subcollections: a cluster-set and a test-set. (The collection used in this case is the 155-query set associated with the Cranfield 424 documents.) In general, the cluster-set provides the queries for clustering; these clusters are then used to generate phase 2 document clusters. When a tree (that is, a hierarchy of documents and centroids) has thus been formed, the test-set queries are used to determine the performance of the particular method used. This simulates the action of an actual information-retrieval system, and makes clear the requirement that the two query-sets be disjoint. (Since one of the hypotheses being tested states that new queries entering a system benefit by the presence of similar queries already processed, it would be unrealistic to test methods of query clustering which allow the "new" queries to be present in the system already.)

Because of the nature of the query collection used, another consideration in this splitting is the authorship of the queries. In a real system, it is unlikely that a given author would submit two queries with similar sets of relevant documents. (If this were the case, relevance-feedback from the results of the previous query should best be used in handling the later query.) The Cranfield queries, however, contain many instances of authors' submitting several queries with similar sets of relevant documents. It is not unreasonable, then, that in splitting such a test collection into two sets of queries, it should be required that author-sets not be broken. That is, if any query of a given author appears in one of the sets, then all queries of that author are put into that set.

With these considerations in mind, the 155 queries are split into two author-consistent sets of 30 (test-set) and 125 (cluster-set) queries each. This is done by generating random numbers between 1 and 155, and including in the test-set those queries whose numbers are drawn at random, as well as all other queries by the same author. Random numbers corresponding to queries already selected are passed by in subsequent drawings. Appendix A, Table A1 gives the results of this splitting, including author number for all queries selected.

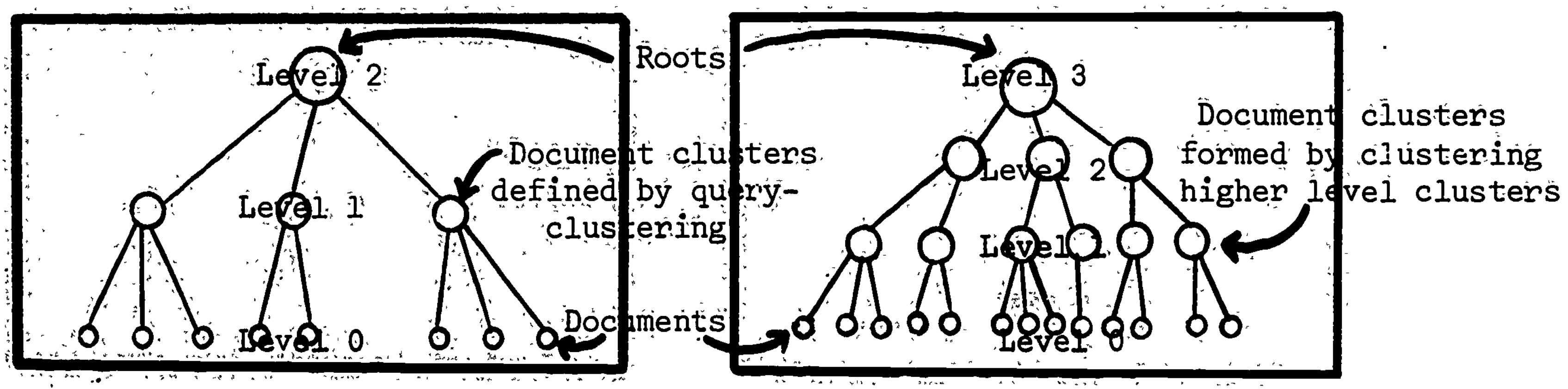
After the generation of the test-set, the cluster-set is formed from the remaining queries. In order to allow the experiment to test the effect of enlarging the base of clustered queries, the 125-item cluster-set is subdivided randomly into sets of 75 and 100 queries, such that the first is a subset of the second. Three cluster-sets — called CS1, CS2, and CS3 — are thus formed — with 75, 100, and 125 queries, respectively — where the increase in size from one to the next is caused only by the inclusion of additional queries. This, too, mirrors the action of a real system, where an increase in queries processed is caused by addition, rather than by reformulation. Table A2 in Appendix A lists the queries included in these three cluster-sets.

B) Phase 1: Clustering the Queries

The three sets of cluster-queries formed by splitting the Cranfield 424 collection queries are clustered using Dattola's algorithm (see [7]). Essential to this algorithm is a specification of the number of clusters desired and the amount of overlap permitted. The results of the experiment in [2] indicate that overlap in such a set of query clusters in greatly magnified when the related document collection is formed — particularly, when relevance decisions are used in phase 2. It is apparent, moreover, that overlap will also be increased by most other implementations of phase 2. Since the overlap obtained in [2] was far too great, it was decided that the query clusters formed here should have no overlap.

Not so easy to answer is the question of how many clusters should be formed. This is a problem in any one-level query-clustering technique. First, as many query-clusters must be formed as the number of desired document clusters. Furthermore, the number of queries to be clustered is generally far less than the number of documents. Thus, the number of clusters cannot be optimal for both the queries and the documents. According to [8], the best number of equal-sized clusters which can be formed from n items is of the order of $\sqrt{n}$. Thus, for the 424 documents of the Cranfield collection, approximately 21 clusters should be made, while the three cluster-sets of queries require, roughly, 9, 10, and 12 clusters, respectively. One solution to the problem is to abandon the single-level hierarchy in favor of a multiple-level tree, where the clusters on level 1 are formed by breaking up the query-clustering-generated clusters on level 2. (see Figure 1.) This method is currently under investigation by Magliozzo and Bodenstein [11].

Because this experiment is not designed to consider such variations in query clustering, a solution other than that mentioned above is desirable. A compromise, therefore, is made between the different optimal numbers of clusters required by the four collections. Dattola's algorithm is asked to provide 15 clusters for each set of queries and documents to be clustered. The 15 clusters of the 424 documents thus generated are later used as a "standard clustering" against which the query-clusterings are compared. (The actual generation of so precise a number of clusters is not a trivial matter, as is discussed in Appendix C.) The results of using Dattola's algorithm to cluster these collections are given in Appendix A, Tables A3 and A4.


Roots
Level 2
Level 1
Level 0
Document clusters defined by query-clustering
Documents
Level 3
Level 2
Level 1
Level 0
Document clusters formed by clustering higher level clusters

(a) Single-level

(b) Multiple-level

Single- and Multiple-Level Clustering

Figure 1

C) Phase 2: Clustering the Documents

In general, the most convenient way to implement phase 2 consists in using relevancy decisions. This is done by assigning each document to the cluster or clusters whose underlying query cluster(s) contain(s) one or more queries to which it is relevant. Unfortunately, this process deals with only part of the documents in the collection, in most cases. Although each document in the Cranfield 424 collection is relevant to one or more queries in that collection, not all of these queries are present in any of the cluster-sets CS1, CS2, and CS3. The documents not assigned to clusters by analysis of relevance may be called "loose documents" (see [8] for a full description of the term), and must be "blended" into the clusters already formed. In all three cases the number of loose documents is rather substantial: 135 in CS1, 89 in CS2, and 54 in CS3. (It should be noted here that, because of the relationship between these three cluster-sets, the loose documents of CS3 are a subset of those of CS2, which are, in turn, a subset of those of CS1.)

Because of the large numbers of loose documents, the method used to incorporate such documents into particular clusters is quite important, and must be the subject of careful scrutiny in any actual use of this method. For the present experiment, however, where the major object is the examination of the results of varying aspects of the clustering scheme other than the blending method, an arbitrary way of assigning loose documents to clusters is chosen. The method consists in correlating each loose document with the 15 centroids of the given cluster-set, and to add each document to the document cluster(s), for which the centroid of the underlying query cluster(s) satisfies one of the following conditions: either the correlation between query centroid and document is 0.250 or higher, or the centroid-document correlation is higher than the correlation with all other query centoids. The reason for this method will become clear when method 2b is discussed below. Results of this assignment — relevancy combined with blending — are given in Appendix A, Table A5.

It is this type of clustering which has been studied previously, and which seems most likely to produce improvements over standard algorithms which do not use query clusters. The effect is to classify documents according to the questions to which they relate, rather than according to similarities in word content. It is, moreover, this method which seems likely to exhibit the most improvement when a larger cluster-base is used. If such a trend could be confirmed, this type of query-clustering would produce a way for an information-retrieval system to "learn" from its past successes, while keeping it from repeating past mistakes. It might also be a way of implicitly altering a system to compensate for changes in terminology, or to anticipate the development of new fields of information. For these reasons, this form of query clustering may have the same type of advantages as document-space modification, a technique examined in Brauen [12].

Type 2 document clusters are formed according to correlations between documents and centroids of the clusters formed in phase 1. In some ways, this might be looked at as a standard clustering algorithm which begins with certain clustcrs already formed, and continues by blending into these the set of documents to be clustered. It might eventually be shown experimentally that using such centroids as a "seed collection" in any of the standard algorithms will produce improved performance from the final clusters.

In the experiment at hand, the procedure is the following for phase 2 using type 2 clusters: all 424 documents are correlated with all 15 query centroids in each cluster set. A document is assigned to any document cluster, for which the centroid of the underlying query cluster satisfies either of two conditions:

a) the centroid and the document correlete at a level of 0.250 or higher;

b) the correlation with the document is higher than that achieved by all other centorids.

Note that this is the same condition under which a loose document is blended into a cluster for method 1. This criterion is chosen, by inspection, to approximate the size and degree of overlap of what might be considered "standard clusters". While such an arbitrary cutoff value is likely to be used in any operational implementaion of this method, the cutoff may, of course, be varied to achieve different clusterings. In appendix A, Table A6 it may be noted that the sizes of the clusters formed vary greatly, from a minimum of 4 by CS1 to a maximum of 85 by CS3. This is clearly an undesirable result, and a method of avoiding it is suggested below. (Varying the cutoff might reduce the problem, but would probably not solve it entirely.)

It should be noted here that method 2 is unlike the previous one in that no loose documents result. This is due, of course, to taking each document individually and assinging it to one or more clusters. The problem of non-uniformly-sized clusters may be solved if the generation of loose documents is permitted. Instead of correlating documents against centroids, it is possible to reverse the process, matching centroids against documents. In this variation of method 2, the top n, say, correlants of each centroid are chosen for inclusion in the document cluster related to that centroid.

Thus, all clusters have the same number of items. By varying the cutoff (n), and imposing additional restrictions on correlations of included documents, it seems likely that interesting results could be obtained. This range of experimentation is not, however, included in the current study.

Finally, the third form of phase 2 is achieved here by correlating all of the 424 documents against the 75, 100, and 125 queries of the three cluster-sets. Documents are assigned to any document cluster whose underlying query cluster contains one or more queries such that either a) that query correlates more highly with that document than any other query, or b) the correlation between the query and the document is 0.250 or more. The motivation for this choice of method is the same as that for the type 2 method, and the same comments apply. In this application, also, a disparity arises in cluster sizes. Table A7 of Appendix A shows that cluster-set CS1 produces both the largest cluster (111 documents) and the smallest (3 documents) generated by method 3.

As before, a reverse strategy may be used which would ensure an even distribution of the documents throughout the clusters (aside from the blending of loose documents).

This method (in either variation) may be regarded as "pre-searching" the document collection in order to make later searches more effective. If, as assumed, many new queries are similar to queries already present in an information-retrieval system, then such a new query should easily find the cluster associated with these similar queries. This method of assigning documents to clusters thus guarantees that the documents in that cluster are those which correlate highly with the old, similar queries (and, thus, hopefully, with the new query).

Both of these last two implementations of phase 2 are inherently more expensive than the first. Relevant documents for a given already-processed query can be easily selected, without the necessity of correlating large numbers of concept vectors. In the case of methcd 2, each document must be correlated against as many centroids as there are query clusters. Method 3 requires a full search of all queries against all documents, although this would be done only once for each document and query. For method 2, a new correlation by all documents would be required with each update of the query clusters. Further analysis of comparative costs of these three methods is possible, but beyond the scope of this paper.

D) Phase 3: Assigning Centroids

The two methods of assigning centroids to document clusters are quite straightforward. In one case, the centroid is taken as that of the underlying query cluster. In the other, it is the standard centroid defined by the documents in the cluster.

Note that using document centroids generally requires an additional series of computations while the use of query centroids does not. This is the case because, as a rule, the process of query-clustering in phase 1 produces the query centroids as a side-effect, at no additional cost. In addition, query centroids tend to be small, taking up less storage space within the machine than document centroids. On the other hand, it may be that document centroids — which contain more of the information about the documents they represent — form a better vehicle for combining documents than qu ry centroids. Even the best clusters will achieve poor performance if the centroids are poorly defined — see Section 4 of this paper — so that this choice, also, is crucial.

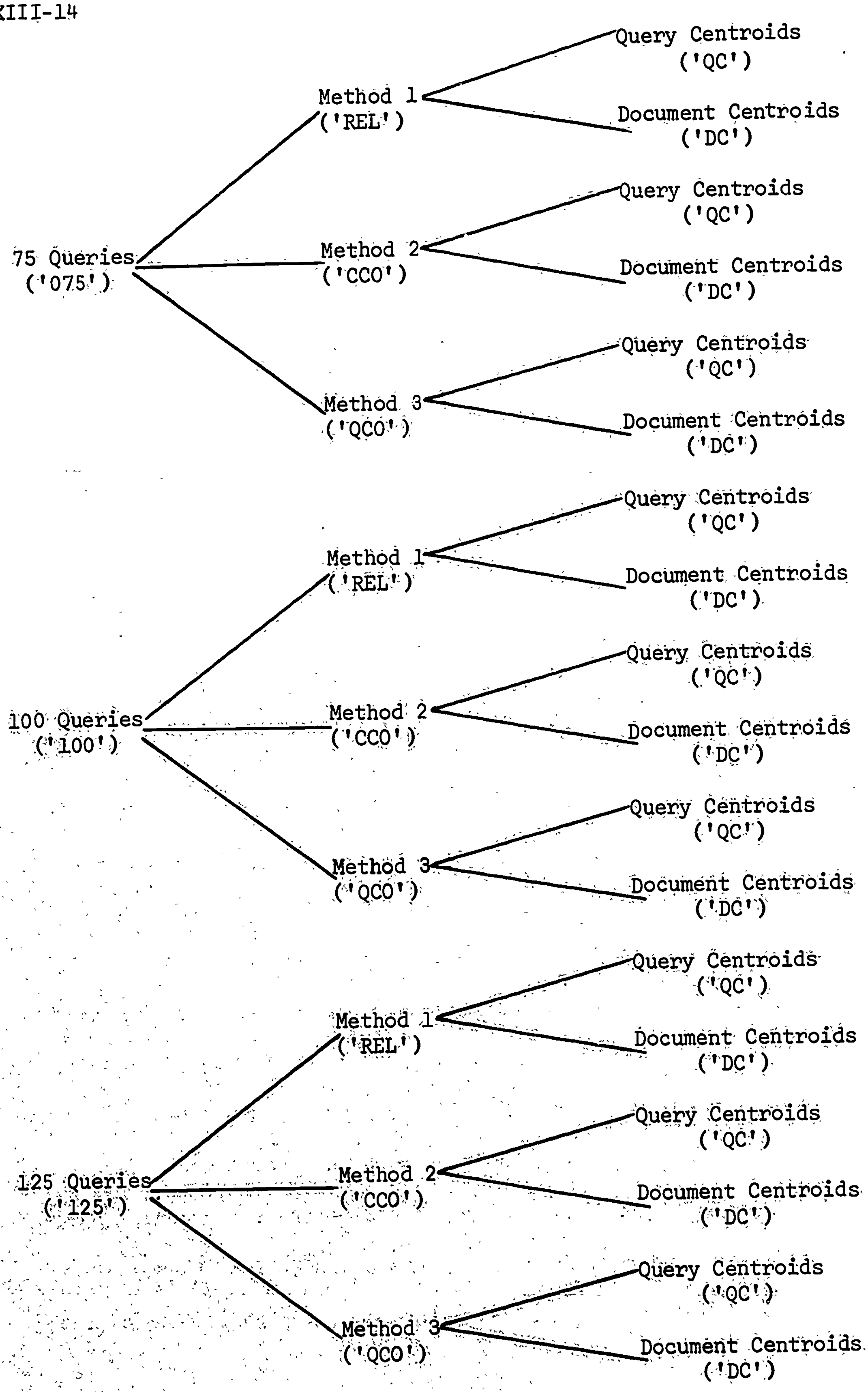
E) Summary

The diagram of Figure 2 indicates the variations used in forming 18 cluster trees from the Cranfield 424 collection. The cluster-collection names used in the rest of this paper may be drived from this diagram by concatenating the three keys describing the particular collection. For example, the collection formed from CS2 (100 queries) using documents assigned by method 2 (centroid correlations) and query centroids in phase 3 is '100CCOQC'.

3. Results

For the most part, the results of this experiment are unexpected. (Graphs of recall-precision values for all 18 clusters are given in Appendix B, together with graphs for the "standard" clusters and a full search using the test-set queries.) Consider, for example, the six comparisons which may be made varying only the number of queries clustered. (The list, including the names introduced in Figure 2, is given in Table 1 below.) Intuitively, the expected ranking in all cases is 125-100-075, best to worst. In only one case out of six, however, is this order achieved: clusters 125QCOQC, 100QCOQC, and 075QCOQC. (See Figure B7.) In four of the five other cases, the clusters using CS2 (100 queries) performed best. (In the remaining case, nnnRELQC, CS3 was best, but CS1 was second-best instead of last.) Moreover, only three of the eighteen cluster-sets produced better results than the "standard" clusters (clusters 100RELDC in Figure B2 and 100CCODC and 075CCODC in Figure B3.) Reference [2] predicts a different result.

The explanations for these results must await further analysis. At present, some observations may be given. It must be noted first that the



Formation of 18 Cluster Collections

Figure 2

1. nnnRELDC	2. nnnRELQC
075RELDC 100RELDC 125RELDC	075RELQC 100RELQC 125RELQC
3. nnnCCODC	4. nnnCCOQC
075CCODC 100CCODC 125CCODC	075CCOQC 100CCOQC 125CCOQC
5. nnnQCODC	6. nnnQCOQC
075QCODC 100QCODC 125QCODC	075QCOQC 100QCOQC 125QCOOC

Six Comparisons of Cluster Groups
by Number of Queries Clustered

Table 1

decisions on the rankings of two or more search results is being made on the basis only of cursory analysis of the recall-precision graphs presented in Appendix B. On that basis, for example, clusters 100RELDC (Figure B2) are being called "better" than the standard clusters of Figure B1, even though the latter rises above the former from recall level .35 onward. A more thorough investigation of these graphs is needed to reach firm conclusions concerning the preferred clustering method.

The possibilities of experimental errors must also be considered. In Appendix C the procedures used in setting up all of these collections are described. Because of the large amount of handwork that went into this stage of the experiment, and because of the lack of complete verification of the resulting lists, it is possible that some degree of error has been introduced in this way.

The final point to be mentioned here concerns the search strategy used. In [2], it is necessary to compensate for unusually large clusters by varying the number of clusters expanded during searching. Since this is not a problem in the present experiment, a fixed number of clusters is expanded in all searches. It is possible that this number does not allow proper representation of the properties of the existing collections. Future searches will include fixed expansions of different values, along with expansions which vary with correlation and number of documents searched. This last consideration seems most promising as an explanation of the results reported here.

4. Principles of Evaluation

In [2] it is suggested that different types of evaluation parameters are needed in full- and clustered-searches. In particular, in a clustered search, it is possible to isolate the cause of good (or bad) performance to one of three areas:

1) Cluster generation

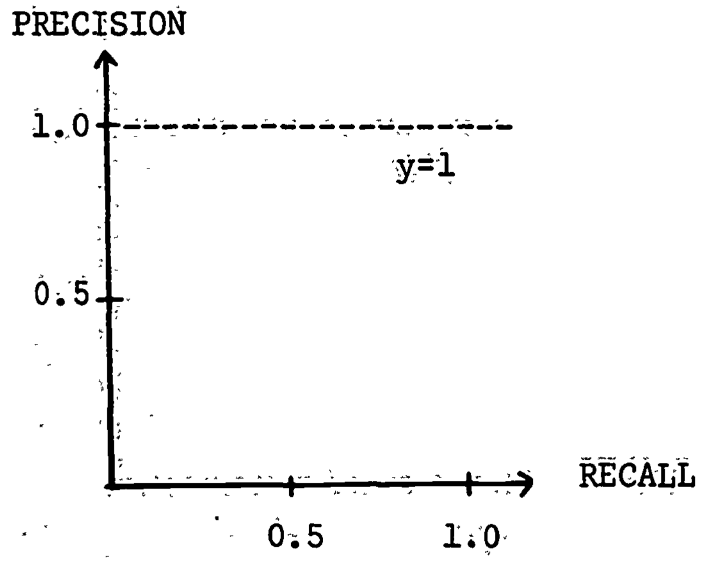
2) Centroid definition

3) Search strategy.

When the search is done without clustering, only the third explanation applies. In addition, the amount of work done in searching a clustered collection may vary from query to query and from one implementation to another, and should also be measured. Although such a measure is sometimes included in the basic performance of a system, it seems clear that the two areas of analysis are quite separate and should be treated as such.

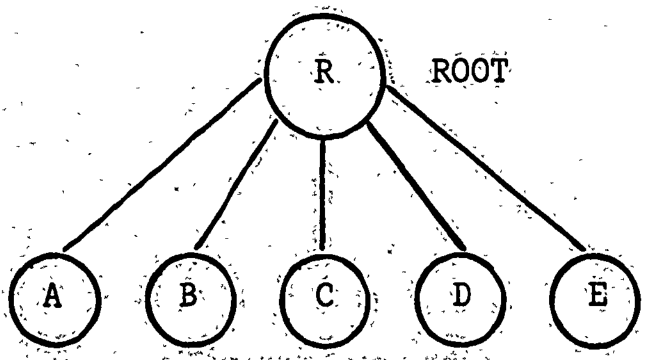
In general, search results are compared to an "optimal" system: one which produces all relevant documents for all queries ranked at the top of the list of retrieved items. In such a system the recall-precision graph achieves a horizontal line at y-intercept 1 (see Figure 3). In the same way, it is psssible to rank the first-level clusters in any clustered collection for each query, in order of "desirability". (Note that the analysis which follows is not directly applicable to multiple-level clusterings. This problem is discussed in [10] and [13].)

Given the configuration of Figure 4, the hoped-for ordering of these clusters before expansion is obviously (high to low number of relevant documents) D-C-A-B=E. Note that two considerations are involved here: First, it is required that the clusters containing the greatest number of relevant documents be ranked highest. Second, between two clusters whose contents of relevant documents are the same, the smaller should be first. The definition of the centroid for each cluster determines how high each cluster ranks, and that definition will determine whether a search does



Optimal Recall-Precision Graph

Figure 3



DOCUMENT CLUSTERS
(Documents Not Shown)

CLUSTER	NO. RELEVANT TO p	NO. IN CLUSTER
A	3	40
B	1	10
C	5	30
D	15	50
E	1	15

n = total documents relevant to p = 25

Clusters with Relevant Documents for Query p

Figure 4

well (or poorly) because the proper clusters are (or are not) expanded.

Consider now a search which orders the clusters of Figure 4 in their "optimal" ranking. This search will, in general, perform less well than a similar one with the clusters of Figure 5, where the clusters are also considered to be ordered correctly before expansion. The reason is clear: in the former case it is necessary to examine 135 documents before all relevant may be included, while with the second group only 80 document correlations must be made. The cluster-set exhibited in the second set possesses as few nonrelevant as possible for those clusters containing all relevant. This property is determined by cluster generation, as apart from centroid generation and search strategy.

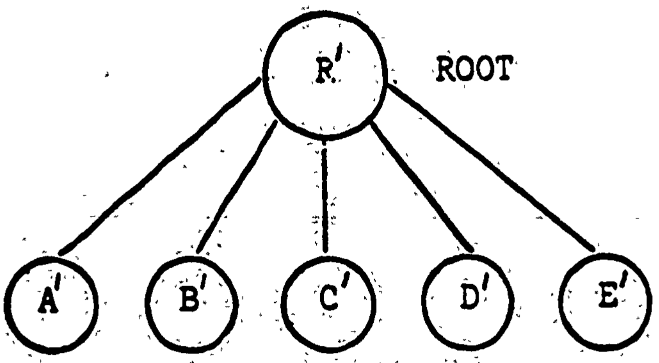
Three parameters are introduced in [2] for dealing with these concepts. They are "aim", "target", and "rejection", defined as follows: Given a query q with n relevant documents. It is assumed that a clustered document collection is to be evaluated according to its clustering success and its achievement in centroid definition. For each number c of clusters expanded,

a) the aim clusters are those c clusters ranked first by whatever correlation procedure is used in ranking clusters, and

b) the target clusters are those c clusters ranked first according to the previously-discussed considerations of a number of relevant contained and size. (For a more precise description, including the question of how documents appearing in more than one cluster are to be treated, see [2].)

The aim value is defined as

$$\frac{\text{number of relevant documents in aim clusters}}{n}$$



DOCUMENT CLUSTERS
(Documents Not Shown)

CLUSTER	NO. RELEVANT TO $\underline{p}$	NO. IN CLUSTER
A'	0	40
B'	0	10
C'	0	30
D'	25	80
E'	0	15

$\underline{n}$ = total documents relevant to $\underline{p}$ = 25

Clusters with Relevant Documents for Query $\underline{p}$
(Second Set)

Figure 5

and the *target value* is

$$\frac{\text{number of relevant documents in target clusters}}{n}.$$

Similarly, *rejection* is defined as

$$\frac{\text{occurrences of rel. documents in target clusters}}{\text{occurrences of rel. documents in all clusters}}.$$

In the measures above, optimal performance is achieved when both the *aim-to-target-ratio* (self-defining) and the rejection are 1 when averaged over all queries. This is a restatement of the definitions of [2], in a more precise form.

The recall-precision graphs are included in Appendix B, and an explanation of the programming tasks is given in Appendix C.

References

[1] V. R. Lesser, A Modified Two-Level Search Algorithm Using Request Clustering, Report No. ISR-11 to the National Science Foundation, Section VII, Department of Computer Science, Cornell University, June 1966.

[2] S. Worona, Query Clustering in a Large Document Collection, Report No. ISR-16 to the National Science Foundation, Section XV, Department of Computer Science, Cornell University, September 1969.

[3] J. D. Broffitt, H. L. Morgan, and J. V. Soden, On Some Clustering Techniques for Information Retrieval, Report No. ISR-11 to the National Science Foundation, Department of Computer Science, Cornell University, June 1966.

[4] R. T. Grauer and M. Messier, An Evaluation of Rocchio's Clustering Algorithm, Report No. ISR-12 to the National Science Foundation, Section VI, Department of Computer Science, Cornell University, June 1967.

[5] R. T. Dattola, A Fast Algorithm for Automatic Classification, Report No. ISR-14 to the National Science Foundation, Section V Department of Computer Science, Cornell University, October 1968.

[6] S. Rieber and V. P. Marathe, The Single Pass Clustering Method, Report No. ISR-16 to the National Science Foundation, Section XIV, Department of Computer Science, Cornell University, September 1969.

[7] R. T. Dattola, Experiments with a Fast Algorithm for Automatic Classification, Report No. ISR-16 to the National Science Foundation, Section XIII, Department of Computer Science, Cornell University, September 1969.

[8] J. J. Rocchio, Document Retrieval Systems — Optimization and Evaluation, Report No. ISR-10 to the National Science Foundation, Harvard University, March 1966.

[9] G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill, Inc., New York, 1968, Ch. 4.

[10] D. Murray, unpublished manuscript.

[11] R. Magliozzo and M. Bodenstein, unpublished manuscript.

[12] R. Brauen, Document Vector Modification, Report No. ISR-17 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

[13] D. Murray, On the Optimal Ranking of Clusters, unpublished manuscript.

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

Appendix A

Tables of Statistics

Query Number	Cran4 Number	Cran14 Number	Author		Query Number	Cran4 Number	Cran14 Number	Author
1	17	67	107		16	64	145	83
2	18	68	107		17	65	146	83
3	19	69	107		18	71	157	17
4	31	97	95		19	72	158	17
5	32	98	95		20	81	175	32
6	33	99	13		21	82	176	32
7	34	100	13		22	84	183	165
8	43	110	80		23	85	184	165
9	44	111	80		24	94	206	147
10	45	112	80		25	104	226	47
11	53	122	25		26	1	274	165
12	54	123	25		27	128	291	17
13	60	137	46		28	131	295	38
14	61	138	46		29	132	296	38
15	62	139	46		30	136	301	32

"Cran4 Number" is the query's number in the 424 collection.
"Cran14 Number" is the query's number in the 1400 collection.
"Author" is the author-number, as given by Cranfield.

Test-Set Queries

Table A1

Qu No.	Clus Set	C4 No.	C14 No.	Qu No.	Clus Set	C4 No.	C14 No.	Qu No.	Clus Set	C4 No.	C14 No.
1	CS1	1	1	43	CS1	99	215	85	CS2	50	119
2	"	2	2	44	"	100	217	86	"	56	131
3	"	3	9	45	"	103	225	87	"	57	132
4	"	6	15	46	"	105	230	88	"	58	135
5	"	7	18	47	"	106	231	89	"	67	148
6	"	10	34	48	"	108	233	90	"	78	148
7	"	12	41	49	"	109	245	91	"	87	190
8	"	13	51	50	"	110	246	92	"	93	205
9	"	14	58	51	"	111	247	93	"	101	218
10	"	15	59	52	"	113	252	94	"	102	224
11	"	21	74	53	"	115	259	95	"	122	273
12	"	23	80	54	"	117	265	96	"	127	288
13	"	25	82	55	"	118	266	97	"	135	299
14	"	28	87	56	"	120	269	98	"	143	332
15	"	29	94	57	"	121	272	99	"	145	335
16	"	30	95	58	"	124	283	100	"	150	352
17	"	37	103	59	"	125	284	101	CS3	5	13
18	"	38	104	60	"	126	285	102	"	8	26
19	"	40	107	61	"	129	293	103	"	24	81
20	"	47	114	62	"	133	297	104	"	27	85
21	"	49	118	63	"	134	298	105	"	35	101
22	"	51	120	64	"	137	306	106	"	39	106
23	"	52	121	65	"	138	314	107	"	41	108
24	"	55	130	66	"	139	315	108	"	42	109
25	"	59	136	67	"	140	316	109	"	48	116
26	"	63	142	68	"	141	321	110	"	70	156
27	"	66	127	69	"	142	323	111	"	73	160
28	"	68	152	70	"	146	336	112	"	79	169
29	"	69	155	71	"	147	347	113	"	83	182
30	"	74	161	72	"	148	348	114	"	89	200
31	"	75	163	73	"	153	356	115	"	90	201
32	"	76	165	74	"	154	360	116	"	107	232
33	"	77	167	75	"	155	365	117	"	112	250
34	"	80	171	76	CS2	4	12	118	"	114	255
35	"	86	187	77	"	9	33	119	"	116	261
36	"	88	196	78	"	11	39	120	"	119	268
37	"	91	202	79	"	16	66	121	"	130	294
38	"	92	204	80	"	20	72	122	"	144	333
39	"	95	209	81	"	22	79	123	"	149	349
40	"	96	210	82	"	26	83	124	"	151	353
41	"	97	211	83	"	36	102	125	"	152	355
42	"	98	214	84	"	46	113				

"C4 No." is the query number in the 424 collection.
"C14 No." is the query number in the 1400 collection.
"CS1", "CS2", and "CS3" mark the beginning of the additional queries for the collections. CS1 consists of queries 1-75; CS2 consists of 1-100; CS3 consists of 1-125.

Cluster-Set Queries

Table A2

Collection Name	CS1	CS2	CS3
No. of queries in collection	75	100	125
Number of clusters formed	15	15	15
Items in largest cluster (no.)	9 (1)	11 (6)	14 (5,10)
% of collection in largest cl.	11	11	12
Items in smallest cluster (no.)	2 (10,12)	3 (8,13,14)	4 (8,11,12)
% of collection in smallest cl.	3	3	3
Repetition factor*	0	0	0

*Defined as the number of total occurrences of documents or queries throughout a clustered collection, divided by the number of distinct items in the collection.

Results of Query Clustering

Table A3

Collection Name	Cran 424 Docs
No. of documents in collection	424
Number of clusters formed	15
Items in largest cluster (no.)	76 (4)
% of collection in largest cl.	18
Items in smallest cluster (no.)	15 (9)
% of collection in smallest cl.	4
Repetition factor*	285

*See note on Table A3.

Results of Clustering the Cranfield 424 Document Collection Using Dattola's Algorithm

Table A4

Collection Prefix	075	100	125
No. of documents in collection	424	424	424
Number of clusters formed	15	15	15
Items in largest cluster (no.)	82 (1)	85 (6)	78 (9)
% of collection in largest cl.	19	20	18
Items in smallest cluster (no.)	5 (10)	14 (13,14)	22 (12)
% of collection in smallest cl.	1	3	5
Repetition factor*	113	178	272

*See note on Table A3.

Results of Type 1 Clustering
(nnnRELxx)

Table A5

Collection Prefix	075	100	125
No. of documents in collection	424	424	424
Number of clusters formed	15	15	15
Items in largest cluster (no.)	80 (2)	81 (15)	85 (14)
% of collection in largest cl.	19	19	20
Items in smallest cluster (no.)	4 (15)	6 (13)	11 (8)
% of collection in smallest cl.	1	1	3
Repetition factor*	64	76	52

*See note on Table A3.

Results of Type 2 Clustering
(nnnCCOxx)

Table A6

Collection Prefix	075	100	125
No. of documents in collection	424	424	424
Number of clusters formed	15	15	15
Items in largest cluster (no.)	111 (1)	107 (6)	89 (14)
% of collection in largest cl.	26	25	21
Items in smallest cluster (no.)	3 (15)	7 (13)	19 (8)
% of collection in smallest cl.	1	2	4
Repetition factor*	171	295	388

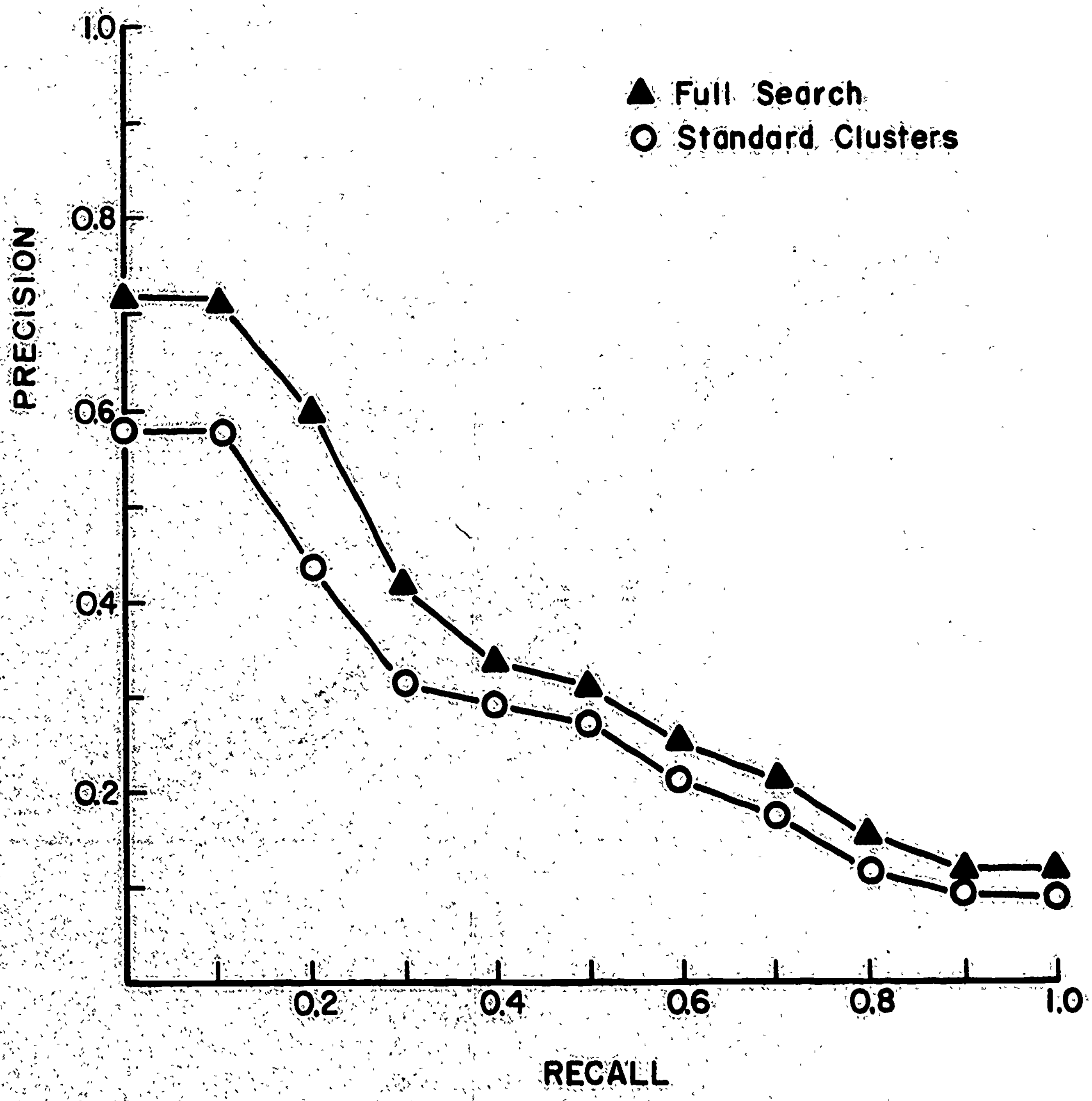
*See note on Table A3.

Results of Type 3 Clustering
(nnnQCOxx)

Table A7

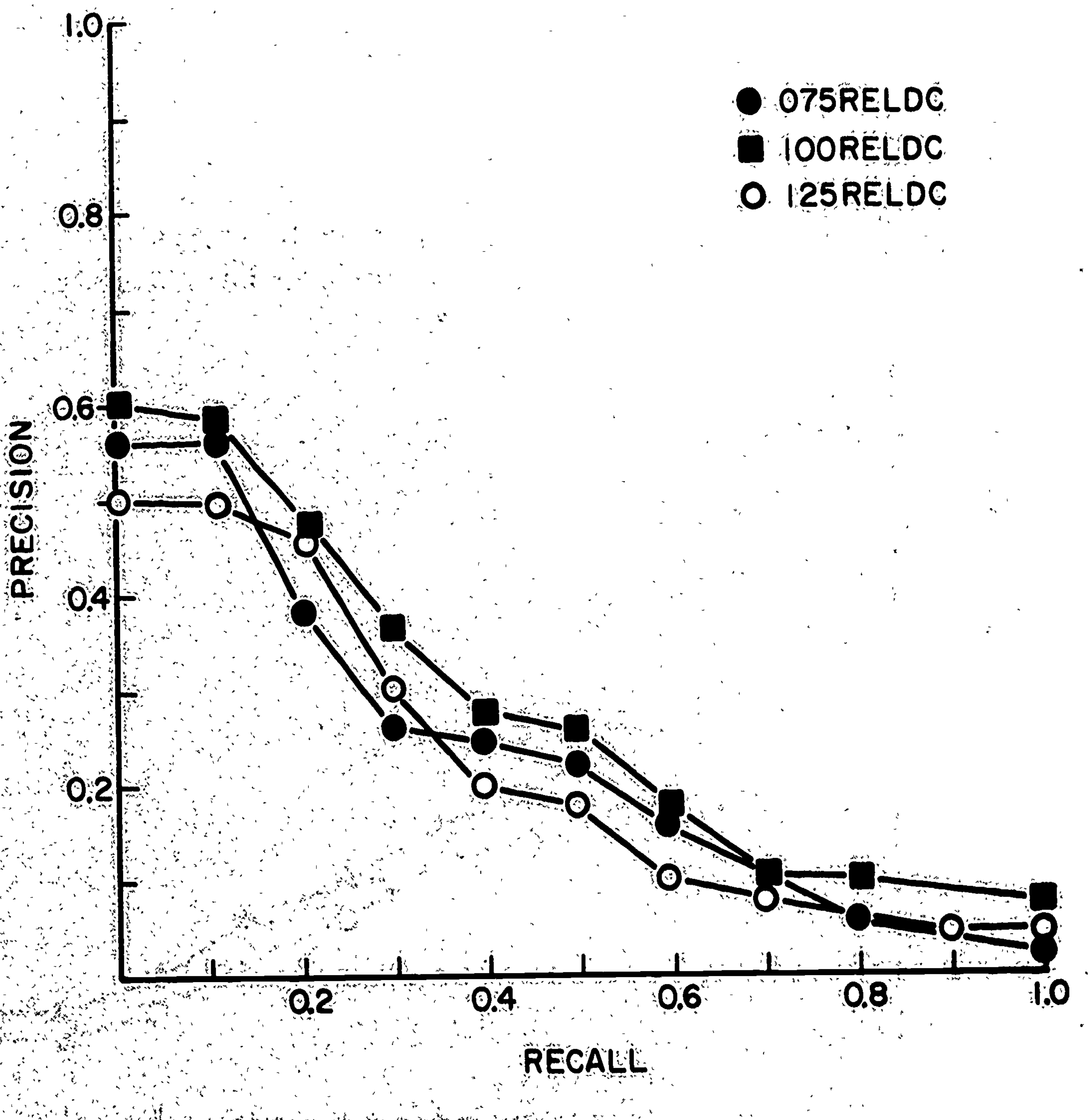
Appendix B

Recall-Precision Graphs of Results



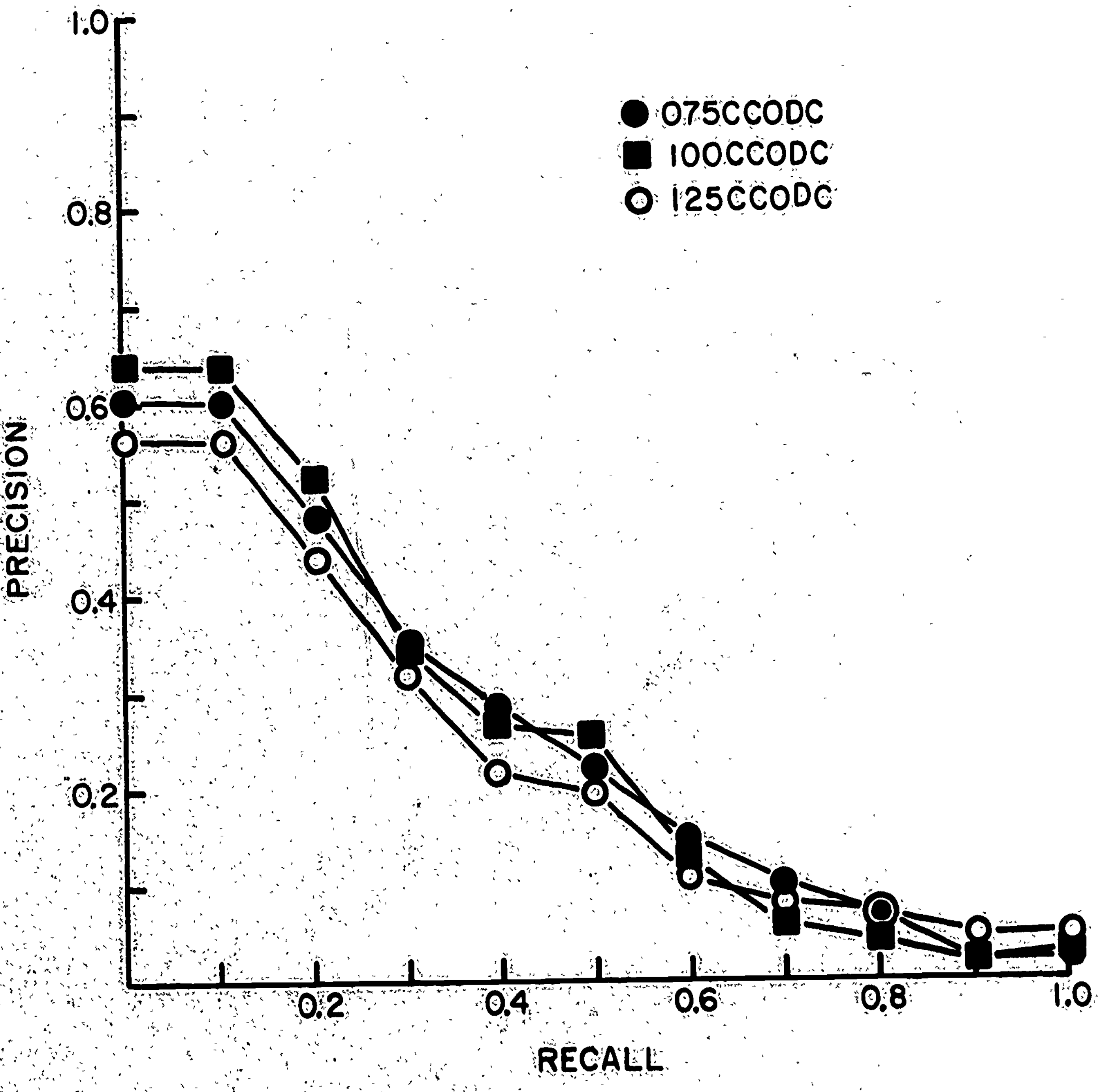
Full Search and Standard Clusters

Figure B1



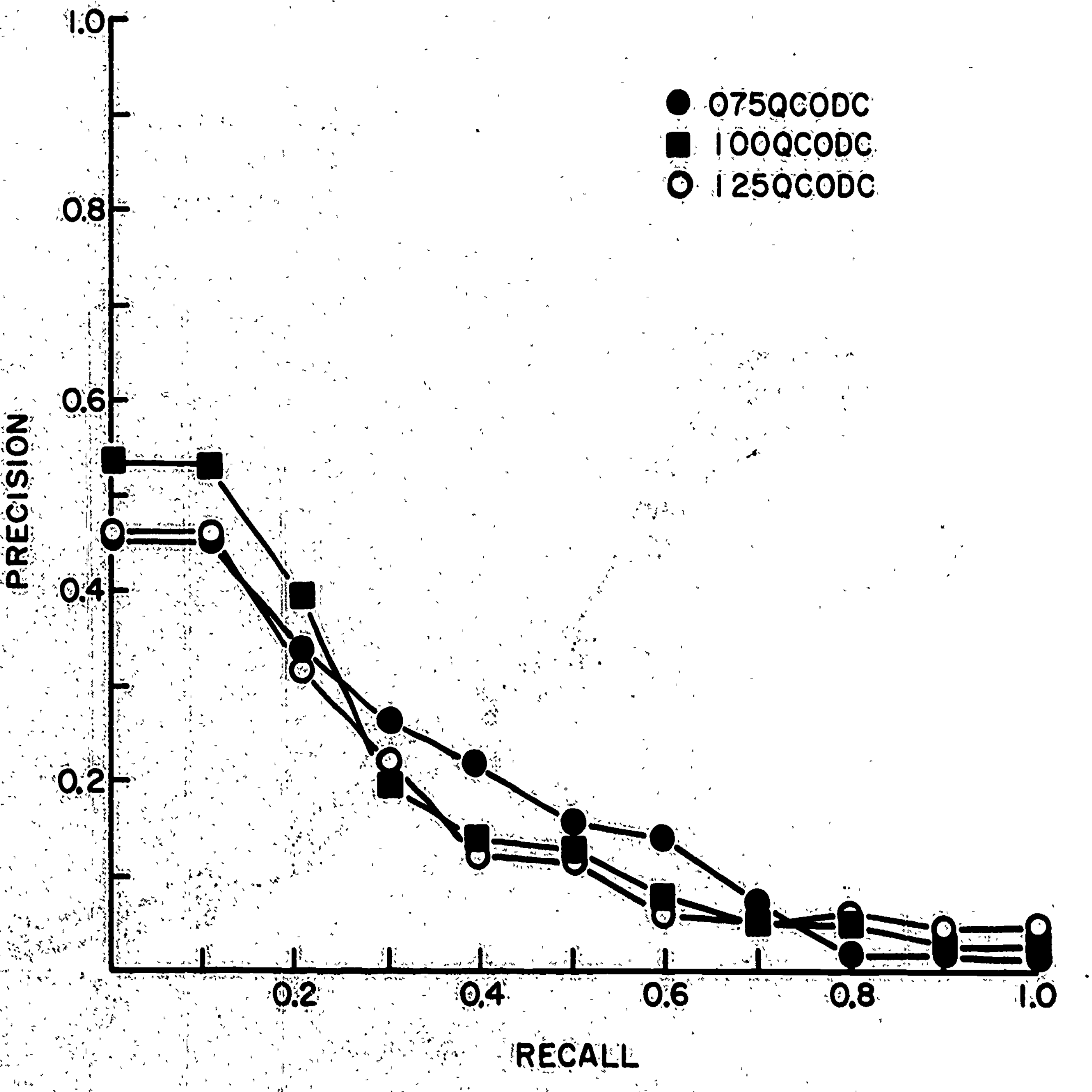
nnnRELDC

Figure B2



nnnCCODC

Figure B3



nnnQCODC

Figure B4

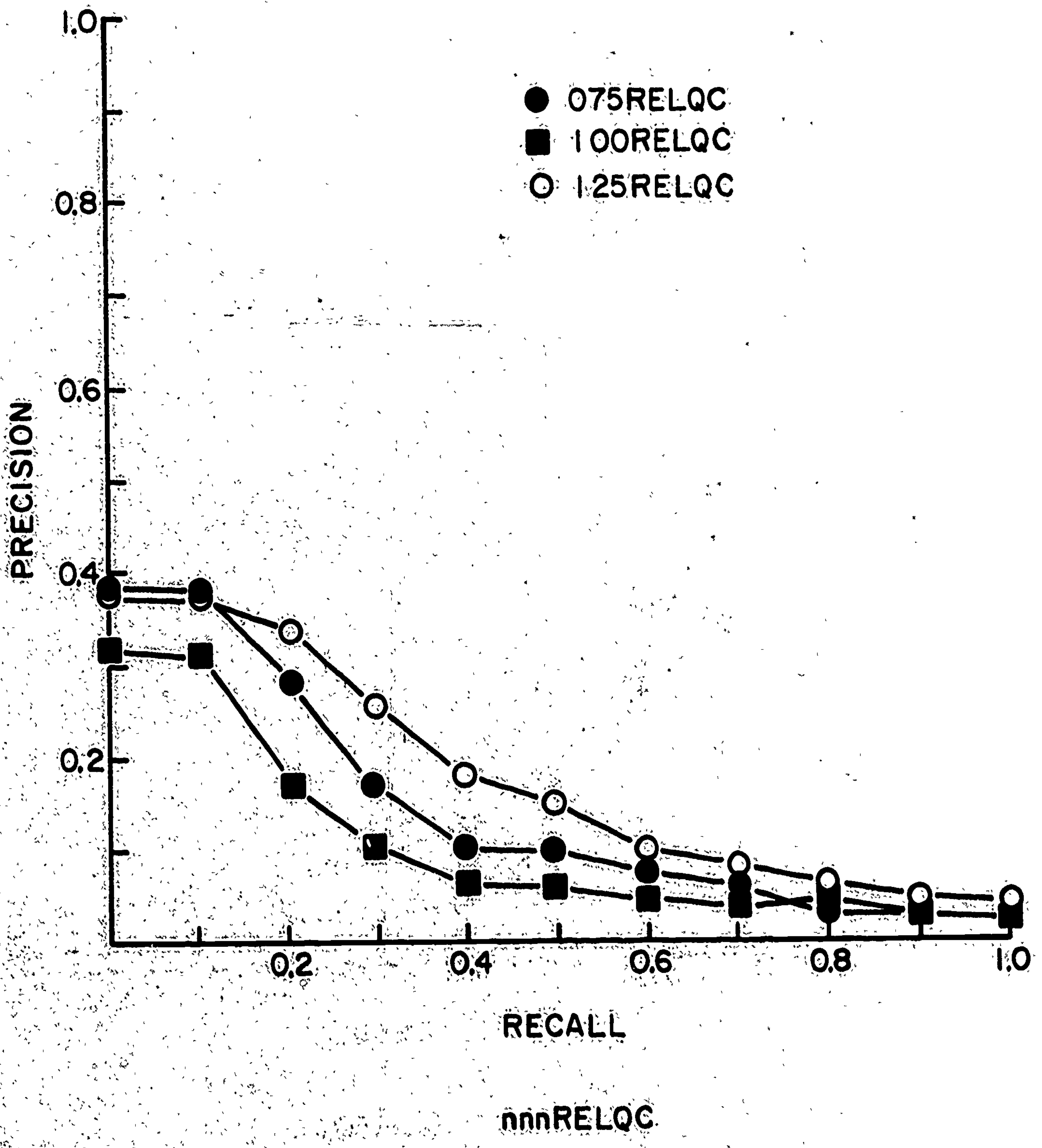
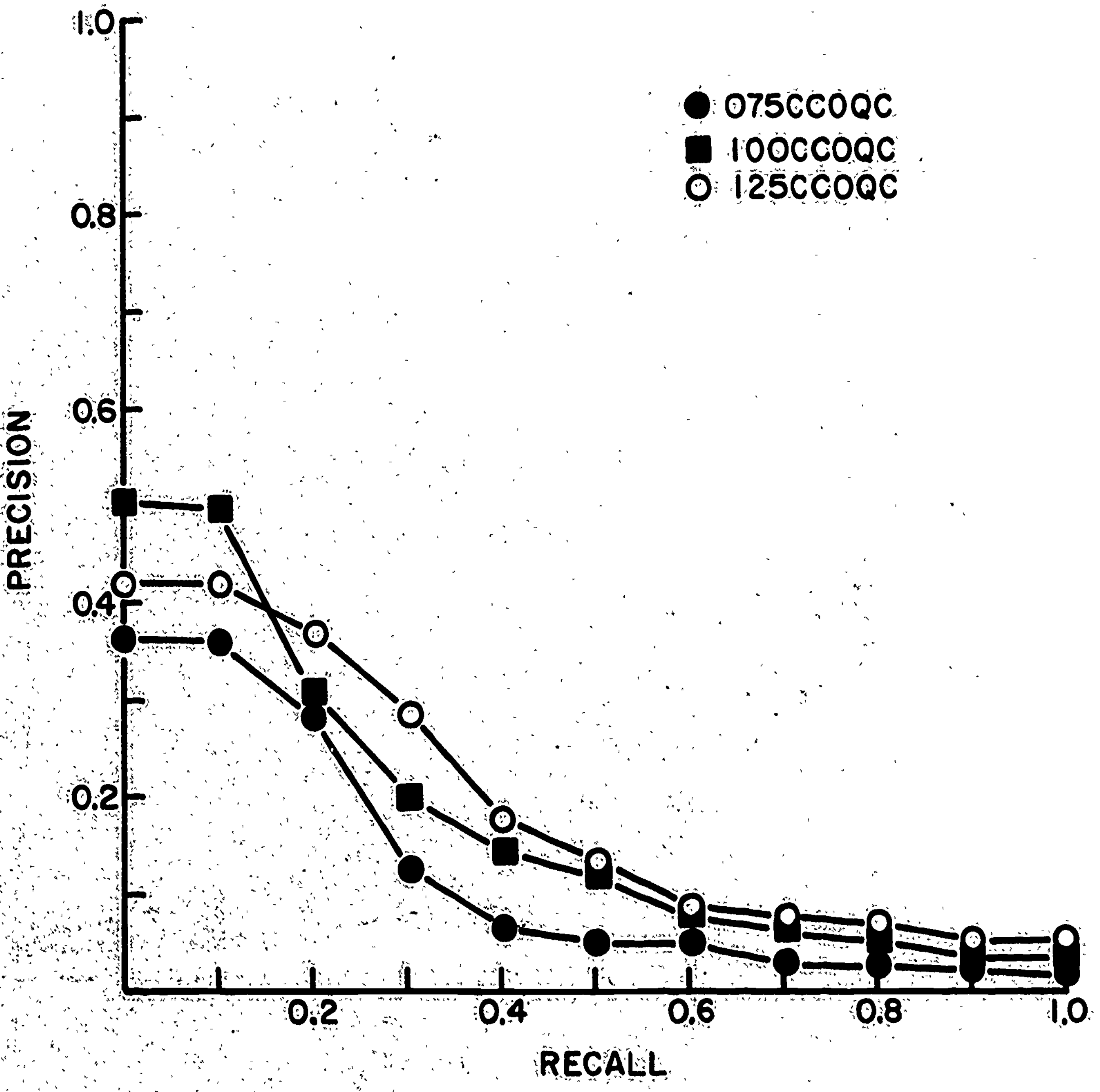
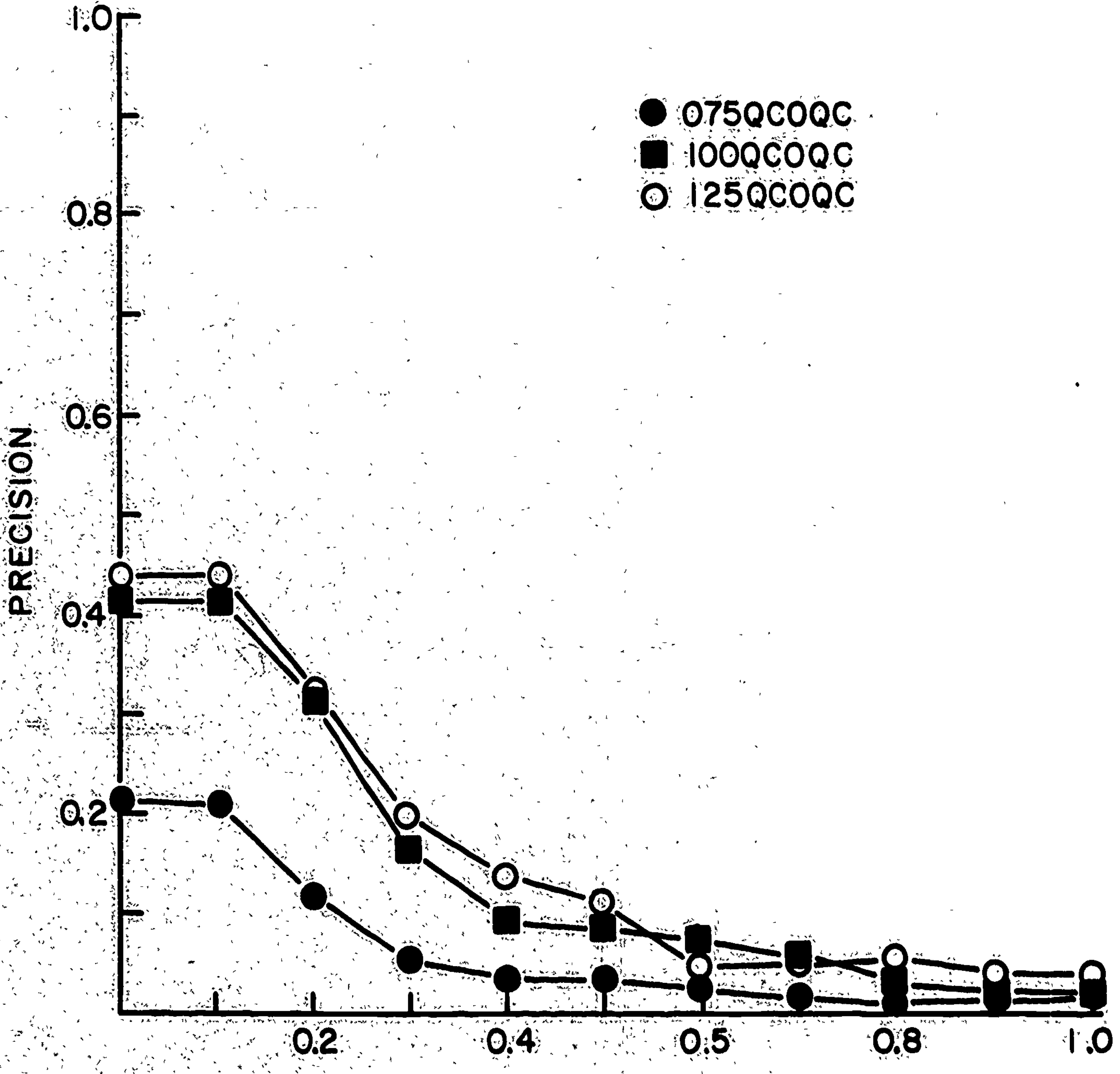

1.0
0.8
0.6
0.4
0.2
PRECISION
● 075RELQC
■ 100RELQC
○ 125RELQC
0.2
0.4
0.6
0.8
1.0
RECALL
nnnRELQC

Figure B5



nnnCCOQC

Figure B6



nnnQCOCQ

Figure B7

Appendix C

The SMART System

In [14] the basic facts about the SMART system implemented at Cornell University are given. At the time of its design, however, no large-scale use of query clustering had been attempted, and none was planned. Therefore, the experiment described in this report required some ad hoc programming and some tiresome hand work. It is hoped that future experimentors with query clustering will benefit from the description given here of the procedures necessary to work within this implementation of SMART in order to carry out such work.

No provision exists in SMART for performing the type of random-number generation and author-set-maintenance described in Section 2. (This is by no means a deficiency of the implementation since such a program is of little general use.) A program was therefore written in FORTRAN to take the information about authors of queries and produce a randomly-selected set subject to the constraints mentioned previously. Such author information is readily available.

At the beginning of the experiment no SMART procedure existed for forming a subcollection of a query or document collection included within the system. Another program was therefore written (also in FORTRAN) to subdivide the Cranfield 424 collection's queries into the four subsets necessary for the experiment (the test-set, and cluster-sets CS1, CS2, and CS3). Recently, however, D. M. Murray has written an addition to the SMART system which performs the necessary subsetting within SMART.

After creating the cluster-sets CS1, CS2, and CS3, Dattola's algorithm (implemented by the SMART procedure DCLSTR) was applied to these collections.

At the same time, the Cranfield 424 documents were processed by DCLSTR. As explained in the text, exactly 15 clusters were required in all cases. It is a property of Dattola's algorithm that the number of clusters produced at any time is a function of the collection used, the random seed specified, and the number of clusters requested. It is thus not possible to predict with accuracy how many clusters will be produced from any set of these parameters. To obtain exactly 15 clusters of the Cranfield documents it was necessary to use several attempts: cluster-set CS2 required 4 tries, cluster-set CS3 needed 7, while only CS1 was successfully divided into 15 clusters in just 1 attempt. A summary of these trial-and-error processes is given in Table C1.

Method 1 of phase 2 was implemented by keypunching the numbers of the documents relevant to each of the queries in each cluster of CS1, CS2, and CS3, and then sorting these three lists with another specially-written (although trivial) program, yielding a listing of the "non-loose" documents. The corresponding loose documents were listed by hand. It was originally assumed that methods 2 and 3 would be done by simple SMART searches, by using the 424 documents as queries against the three sets of centroids and the three sets of queries. This large number of "queries" proved unworkable in the system, and another program modification was required.

After the definitions for all 9 cluster sets were completed, it remained to generate 18 centroid sets, and unite these two parts of the ultimate collections. A SMART routine called CRDCEN has as its purpose this exact function. The experiment was delayed, however, by the necessity to keypunch a great deal of information from the previous tabulations. In it recommended that when a program is written to perform

Cluster Set	Attempt	Seed	No. Clusters Requested	No. Clusters Received
CS1	1	.12345	15	15*
CS2	1	.12345	15	12
	2	.54321	15	13
	3	.54321	16	14
	4	.54321	17	15*
CS3	1	.12345	15	14
	2	.54321	15	13
	3	.12345	16	14
	4	.12345	17	14
	5	.12345	18	16
	6	.54321	18	16
	7	.54321	17	15*
424 docs	1	.12345	15	11
	2	.12345	18	15*

*satisfactory clusters

Parameters Used in Generating Clusters
with Dattola's Algorithm

Table C1

the required tabulation automatically, this should be done with a v.:ew to obtaining the data and format required by CRDCEN, the process to which the results will eventually be passed.

Eventaully, the entire process should be made a part of the SMART system to be invoked like any standard clustering algorithm.