V. A Fast Algorithm for Automatic Classification

R. T. Dattola

Abstract

Several different methods exist for classifying the elements of a file into groups based on similarities in the attributes of the elements. In information retrieval, the elements are frequently documents, and the attributes are words or concepts characterizing the documents.

Most known procedures are based on the construction of similarity matrices specifying the pairwise similarities between each pair of elements. Such n-square procedures (for n elements) are expensive to carry out in terms of time and memory space. In the present study, a classification process of order n log n (for n elements) is described, and convergence proofs are given. Possible applications to information retrieval are discussed.

1. Introduction

Many methods exist for ordering or classifying the elements of a file. The elements are usually clustered into groups based on the similarities of the attributes of the elements. In information retrieval, the elements are frequently documents, and the attributes are words or concepts characterizing the documents. Classification of document files may be divided into two basic categories:

a) an a priori classification already exists and each document is placed into the cluster whose centroid is most similar to that document;

b) no a priori classification is specified and clusters are formed only on the basis of similarities among documents.

Classification schemes that fall into the first class are very common and often involve manual methods. For example, new acquisitions of a library are classified by placing them into the clusters of a standard, a priori classification. Problems of the second type are usually more difficult to handle, and automatic or semi-automatic methods are often used. Methods of this type are widely used in statistical programs, but the number of elements in the file is limited to several hundred, or at most, a few thousand items. In information retrieval applications, the number of elements may approach several hundred thousand or even a million documents, as in the case of a large library. In the present study, a method is described which is suitable for classification of very large document collections.

2. The N^2 Problem

Current methods of automatic document classification usually require the calculation of a _similarity matrix_. This matrix specifies the correlation, or similarity, between every pair of documents in the collection. Thus, if the collection contains N documents, N^2 computations are required for calculation of the similarity matrix.* This immediately

*Very often, the similarity matrix is symmetric, so the number of computations is reduced to $N^2/2$.

poses two serious problems: the storage space necessary to store the matrix increases as the square of the number of documents, and the time required to calculate the matrix also increases quadratically. Fortunately, document-document similarity matrices are normally only about ten percent dense, and only the non-zero elements need be stored [1]. However, as N increases, auxiliary storage must eventually be used, and although this solves the space problem, it also magnifies the time problem.

To illustrate the magnitude of this problem, suppose that it takes one hour of computer time to classify a one thousand document collection. Then for $N = 10^4$, the time is approximately one hundred hours, and for $N = 10^6$, the time needed is about 120 years! The classification scheme described in this paper is an adaptation of the one proposed by Doyle, and the time required is of the order of N log N [2]. For example, assuming the logrithm has base 10, and the time required for a one thousand document collection is again one hour, then for $N = 10^4$ the time is 13 hours, and for $N = 10^6$, the time required is about 83 days.

3. Doyle's Algorithm

The N^2 problem is avoided in this classification scheme, because a similarity matrix is never computed. Assume the document set is arbitrarily partitioned into m clusters, where S_j is the set of documents in cluster j. Associated with each set S_j a corresponding concept vector C_j and frequency vector F_j are associated. The concept vector consists of all the concepts occurring in the documents of S_j, and the frequency vector specifies the number of documents in S_j in which each concept occurs.

Every concept in C_j is assigned a rank according to its frequency; i.e., concepts with the highest frequency have a rank of 1, concepts with the next highest frequency receive a rank of 2, etc. Given an integer b (base value), every concept in C_j is assigned a rank value equal to the base value minus the rank of that concept. The vector of rank values is called the profile P_j of the set S_j. Fig. 1 illustrates the concept and frequency vectors, and the corresponding profiles for a sample document collection.

Starting from a partition of the document set into m clusters, the profiles are generated as described. Every document d_i in the document space is now scored against each of the m profiles by a scoring function g, where $g(d_i, P_j)$ = the sum of the rank values of all the concepts from d_i which occur in C_j. Fig. 2 shows the results of scoring the documents in the sample collection against the profiles from Fig. 1.

Given a cut-off value T, a new partition of the document set into m+1 clusters is made by the following formula:

$$S'_j = \left\{ d_i \mid g(d_i, P_j) \geq g(d_i, P_k) \text{ and } g(d_i, P_j) \geq T, \text{ for } k = 1, \ldots, m \right\} .$$

Thus, S'_j consists of all the documents that score highest against profile P_j, provided that the score is at least as great as T. In cases where a document scores highest against two or more profiles, say $P_{r_1}, \ldots, P_{r_n}$, the following tie-breaking rule is used:

d_1	d_2	d_3	d_4	d_5	d_6	d_7
c_1	c_1	c_1	c_1	c_1	c_3	c_6
c_2	c_2	c_7	c_2	c_8		c_8
c_5	c_4	c_8	c_3			
	c_5		c_5			

a) Documents

S_1	C_1	F_1	P_1	S_2	C_2	F_2	P_2	S_3	C_3	F_3	P_3
d_1	c_1	3	5	d_2	c_1	2	5	d_6	c_3	1	5
d_3	c_2	1	3	d_4	c_2	2	5	d_7	c_6	1	5
d_5	c_5	1	3		c_3	1	4		c_8	1	5
	c_7	1	3		c_4	1	4				
	c_8	2	4		c_5	2	5				

b) Initial Clusters, Profiles, and Frequencies

Construction of Profiles from Document Partition
(base value = 6)

Fig. 1

Document	Profile of Highest Score	Score
d_1	2	15
d_2	2	19
d_3	1	12
d_4	2	19
d_5	1	9
d_6	3	5
d_7	3	10

a) Document Scoring

S'_1	S'_2	S'_3	L
d_3	d_1	d_7	d_5
	d_2		d_6
	d_4		

b) Resulting Clusters

One Iteration of Classification Algorithm
(cut-off = 10)

Fig. 2

if $d_i \varepsilon S_j$ and,

a) $j = r_k$, $1 \leq k \leq n$, then d_i is assigned to S'_{r_k};

b) $j \neq r_k$ for $1 \leq k \leq n$, then d_i is arbitrarily assigned to S'_{r_1}.

Those documents which do not fall into any of the m clusters S'_j are called <u>loose documents</u>, and they are assigned to a special class L. The process is now repeated after replacing P_j by P'_j. The iteration continues until P_j satisfies the <u>termination condition</u>, which states that $P'_j = P_j$ for $j = 1,\ldots,m$; i.e., the profiles are unchanged after two consecutive iterations.

4. Satisfaction of Termination Condition

A) Non-convergence of Doyle's Algorithm

Doyle's algorithm as described is not guaranteed to terminate. To illustrate this, consider the following document collection:

d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
1	1	3	3	1	2	2	2	1
2	5	7	4	5	3	3	4	2
3	6	8	7	6	4	4	5	4
4	11	9	8	7	9	7	6	5
5	12	12				8	7	6
11						9	8	7
						10	10	10

Let $S_1 = \left\{d_1 - d_6\right\}$, $S_2 = \left\{d_7 - d_9\right\}$, and let P_1 = profile of S_1, and P_2 = profile of S_2. The two profiles are as follows (base value = 7):

$d_1 - d_6$			$d_7 - d_9$		
Concept	Frequency	Profile	Concept	Frequency	Profile
1	3	5	1	1	4
2	2	4	2	3	6
3	4	6	3	1	4
4	3	5	4	3	6
5	3	5	5	2	5
6	2	4	6	2	5
7	3	5	7	3	6
8	2	4	8	2	5
9	2	4	9	1	4
10	0	-	10	3	6
11	2	4	11	0	-
12	2	4	12	0	-
13	0	-	13	1	4

Now assume that $T = 0$, and partition the document set by the formula

$$S_1' = \left\{ d_i \middle| g(d_i, P_1) \geq g(d_i, P_2) \right\} \text{ and } S_2' = \left\{ d_i \middle| g(d_i, P_2) \geq g(d_i, P_1) \right\} .$$

The results are summarized in the following table:

	$g(d_i, P_1)$	$g(d_i, P_2)$
d_1	29	25
d_2	22	14
d_3	23	19
d_4	20	21
d_5	19	20
d_6	19	20
d_7	28	37
d_8	27	39
d_9	28	42

Therefore, $S_1' = \left\{ d_1 - d_3 \right\}$ and $S_2' = \left\{ d_4 - d_9 \right\}$.

According to Doyle's algorithm, P_1 is replaced by P_1' and P_2 by P_2'. The new profiles are:

$d_1 - d_3$			$d_4 - d_9$		
Concept	Frequency	Profile	Concept	Frequency	Profile
1	2	6	1	2	3
2	1	5	2	4	5
3	2	6	3	3	4
4	1	5	4	5	6
5	2	6	5	3	4
6	1	5	6	3	4
7	1	5	7	5	6
8	1	5	8	3	4
9	1	5	9	2	3
10	0	-	10	3	4
11	2	6	11	0	-
12	2	6	12	0	-
13	0	-	13	1	2

Now the document set is again partitioned and the results are:

	$g(d_i, P_1)$	$g(d_i, P_2)$
d_1	34	22
d_2	29	11
d_3	27	17
d_4	21	20
d_5	22	17
d_6	21	18
d_7	31	32
d_8	31	33
d_9	32	34

Therefore, $S_1' = \left\{d_1 - d_6\right\}$ and $S_2' = \left\{d_7 - d_9\right\}$. These are the original sets, so that the algorithm will never terminate for this example.

B) Termination of Modified Algorithm

Although Doyle's algorithm is not guaranteed to terminate, Needham proved that similar types of iterative methods are guaranteed to terminate in a finite number of steps [3]. A small change in Doyle's method produces an algorithm that is guaranteed to terminate. The modification occurs after the calculation of the S_j'. Instead of automatically replacing the old P_j by P_j', the following condition must also be satisfied:

$$\sum_{i \varepsilon S_j'} g(d_i, P_j') > \sum_{i \varepsilon S_j'} g(d_i, P_j)$$

If the above condition is not satisfied, P_j is left unchanged.

Before proving that this new algorithm is guaranteed to terminate, it is desirable first to make the algorithm more general by allowing overlap between the clusters. The following theorem proves the termination of a method which allows overlapping clusters.

Theorem: Let the subscript n designate the nth iteration. Let D represent the document space and let $P_{0,1}, \ldots, P_{0,m}$ represent m initial profiles corresponding to an arbitrary distribution $S_{0,1}, \ldots, S_{0,m}$ of documents in D.

Given a cut-off value T, the nth iteration is defined as follows:

1. Generate the sets $S_{n,1}, \ldots, S_{n,m}$ and L_n by

$$S_{n,j} = \left\{ d_i \mid g(d_i, P_{n-1,j}) \geq T \right\}$$
$$L_n = \langle \text{ loose documents } \rangle$$

2. Let $P_{n,j} = \begin{cases} P_{n,j} \text{ if } \sum\limits_{i \varepsilon S_{n,j}} g(d_i, P_{n,j}) > \sum\limits_{i \varepsilon S_{n,j}} g(d_i, P_{n-1,j}) \\ P_{n-1,j} \text{ otherwise} \end{cases}$

This algorithm is guaranteed to terminate in a finite number of iterations, where termination occurs when $P_{n,j} = P_{n-1,j}$ for all j.

Proof: Extend the document space D to a new document space D# containing m distinguishable copies of every document in D. Also, add the condition that $S_{n,j}$ can never contain more than one copy of each document. Clearly, any $S_{n,j}$ defined on D# in this manner can also be represented on D as defined in the theorem. Conversely, any $S_{n,j}$ defined on D can also be represented on D# as defined above. Thus, it suffices to prove the theorem on D# under the added condition.

Define a function F_n, which will be shown to be monotone increasing in n, by the following:

$$F_n = \sum_{j=1}^{m} F_{n,j} + T \cdot Z_n, \text{ where}$$

$$F_{n,j} = \sum_{i \varepsilon S_{n,j}} g(d_i, P_{n-1,j}) \text{ and}$$

Z_n = number of documents in L_n.

After step 2 of the iteration, F_n is replaced by F'_n, where

$$F'_{n,j} = \sum_{i \varepsilon S_{n,j}} g(d_i, P_{n,j}) .$$

If for any j, $P_{n,j} \neq P_{n-1,j}$, then $F'_{n,j} > F_{n,j}$* and therefore $F'_n > F_n$. If termination occurs; i.e., $P_{n,j} = P_{n-1,j}$ for all j; then $F'_n = F_n$. For the n+1th iteration,

$$F_{n+1} = \sum_{j=1}^{m} F_{n+1,j} + T \cdot Z_{n+1}, \text{ where}$$

$$F_{n+1,j} = \sum_{i \varepsilon S_{n+1,j}} g(d_i, P_{n,j}).$$

Consider the relation between the contribution of d_i to F'_n and F_{n+1}, and note that each d_i (where copies of a document are distinct) contributes once and only once to both F'_n and F_{n+1}. This relation is summarized in the following table:

document d_i	relation between contribution of d_i to F_{n+1} and F'_n
a) was assigned to $S_{n,j}$ and now	
1. to $S_{n+1,j}$ (d_i did not change clusters)	=
2. to $S_{n+1,k}$, $k \neq j$ (d_i did change clusters)	> ($g(d_i, P_{n,j}) < T$; otherwise d_i would be in $S_{n+1,j}$. Also, $g(d_i, P_{n,k}) \geq T$)
3. to L_{n+1}	> ($g(d_i, P_{n,j}) < T$. Now d_i contributes $1 \cdot T = T$ to F_{n+1})
b) was assigned to L_n and now	
1. to L_{n+1}	= (contributes T to both)
2. $S_{n+1,j}$	$\geq$ (gave T for F'_n, and now gives $g(d_i, P_{n,j}) \geq T$ for F_{n+1})

*This statement is not necessarily true in Doyle's algorithm.

Therefore, $F_{n+1} \geq F'_n$. If $P_{n,j} = P_{n-1,j}$ for all j, then from a) 1 and b)1 $F_{n+1} = F'_n$. Therefore, if the termination condition is satisfied, then $F_{n+1} = F_n$. On the other hand, if $F_{n+1} = F_n$, then $F_n = F'_n$, which occurs only when termination occurs.

Thus, F_n is a monotone increasing function, where $F_{n+1} = F_n$ if the termination condition is satisfied. Given m and T, F depends only on the distribution of the documents of D# in $S_1, \ldots, S_m$. Since there are only a finite number of distributions, there are only a finite number of values for F. Therefore, at some iteration n, F_{n+1} must equal F_n.

5. Implementation

The algorithm described in the preceding section is not implemented. Instead, experiments are performed using an algorithm which differs from the preceding one in four important respects:

1. the extra condition necessary for convergence that is mentioned in section 4B is not implemented; i.e., P_j is always replaced by P'_j;
2. termination occurs when $S^*_{n,j} = S^*_{n+1,j}$ for all j, where $S^*_{n,j}$ is the subset of $S_{n,j}$ consisting of all those documents that score <u>highest</u> against profile $P_{n,j}$;
3. let $H_{n,i} = \max\limits_{1 \leq j \leq m} (g(d_i, P_{n,j}))$, and define $S_{n,j}$ as

$$S_{n,j} = \left\{ d_i \,\middle|\, g(d_i, P_{n-1,j}) \geq T_{n-1,i} \right\}, \text{ where}$$

$$T_{n-1,i} = \begin{cases} H_{n-1,i} - \left[a \cdot (H_{n-1,i} - T) \right], & \text{if } H_{n-1,i} > T, \text{ where } 0 \leq a \leq 1 \\ T \text{ otherwise} \end{cases}$$

4) if any $S_{n,j}$ contains fewer than 2 documents, then $S_{n,j}$ is eliminated, thereby reducing the number of clusters by one.

The advantages of this method over the one defined in the theorem are discussed in the present section; the disadvantage is, of course, that termination is not guaranteed. To show this, note that conditions 1 and 2 above are equivalent to the termination condition in Doyle's algorithm, since in Doyle's method $P_{n,j}$ always corresponds to the new partition $S_{n,j}$, and $S_{n,j} = S^*_{n,j}$ (no overlap is allowed). Also, if a = 0 in condition 3, then $T_{n-1,i} = H_{n-1,i}$. Thus, only those documents d_i that score highest against $P_{n-1,j}$, where $H_{n-1,i} \geq T$, are assigned to $S_{n,j}$. Therefore, with a = 0 this method is equivalent to Doyle's algorithm.

The first two modifications are implemented to improve the efficiency of the program. Although convergence is no longer guaranteed, all the experiments tried so far have in fact always terminated. Programs without these two modifications run about twice as slow. Also, in cases where the overlap is not too high ($S^*_{n,j} \approx S_{n,j}$), the new termination condition is usually equivalent to the one used in the theorem. That is, when $S^*_{n,j} = S^*_{n+1,j}$, then very often $S_{n,j} = S_{n+1,j}$.

The third modification does not improve efficiency, but it allows a more flexible, and intuitively, a more desirable method for creating overlap. The algorithm described in the theorem assigns a document d_i to a cluster $S_{n,j}$ iff $g(d_i, P_{n-1,j}) \geq T$. This has two major disadvantages:

1) the overlap cannot be increased independently of the number of loose documents; <u>increasing</u> the overlap by lowering T in general <u>decreases</u> the percentage of loose documents;

2) the difference between d_i's highest score and d_i's second highest score is ignored; e.g., if $T = 50$, $g(d_i, P_1) = 200$, and $g(d_i, P_2) = 50$, then d_i is assigned to both S_1 and S_2.

The first problem decreases the flexibility of the algorithm, since the amount of overlap and percentage of loose documents cannot be varied independently. The example in the second part illustrates the other problem. It seems desirable that a document should be assignable to two or more clusters when it scores equally (or almost equally) as high against all of them. The previous method does not take this fact into account. In the new algorithm, documents are assigned to more than one cluster on the basis of how close the score is to the highest score, relative to the cut-off value T. The parameter a determines how close to the highest score the other scores must be. When $a = 0$, no overlap occurs, while $a = 1$ generates the maximum amount of overlap.*

The last modification increases the efficiency of the program, and also avoids forming clusters around documents which should be classified as loose. When $S_{n,j}$ contains only one document, and that document is contained in no other clusters, then it has the same status as a loose document.

*With $a = 1$, the formula reduces to $T_{n-1,i} = T$; hence, it is the same definition of $S_{n,j}$ as in the theorem.

6. Experimental Results

The algorithm described in section 5 is used to cluster the 82 document ADI collection and the 200 document Cranfield word stem collection. The results of the classification indicate three important problems:

a) the scoring function g tends to give higher scores to documents containing a larger number of concepts; thus, many of the documents containing very few concepts are classified as loose;

b) the documents do not move freely enough from one profile to another; i.e., the final clusters are quite similar to the initial ones;

c) the initial clusters cannot be chosen arbitrarily.

A) The Scoring Function

The first problem is due to the fact that g scores a document d_i against a profile P_j by simply adding up the rank values of all the concepts in d_i which appear in P_j. If d_i contains a larger number of concepts than d_k, the chances are greater for d_i to receive a higher score. Fig. 3 is a plot of the score of the document against its final profile vs. the number of concepts in the document for one of the ADI runs. Although there are a few exceptions, the graph indicates that the documents with a larger number of concepts generally receive higher scores. In fact, the average number of concepts in a loose document is 11, while the average number of concepts per document for the entire collection is 20.

The solution to this problem is to weight the score inversely by the number of concepts in the document. The obvious answer is to divide

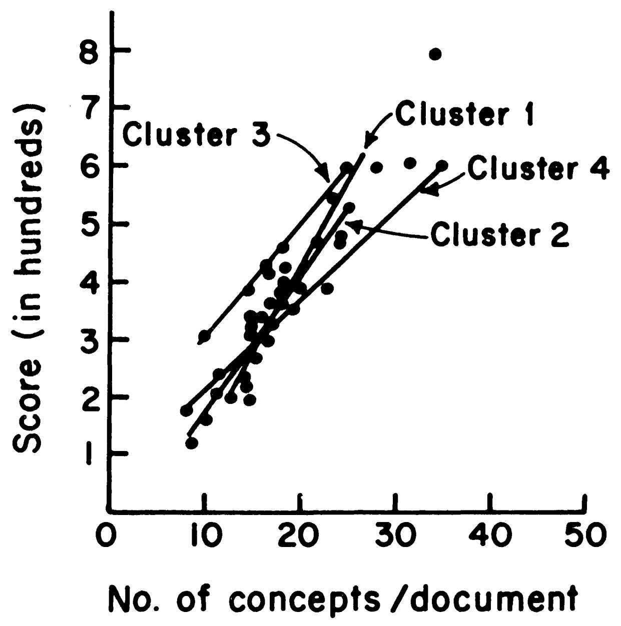

Illustration of Initial Scoring Function

Fig. 3

the score by the number of concepts, but this overcompensates and gives many of the smaller documents the highest scores. Dividing by the square root of the number of concepts in the document does not solve the original problem; i.e., larger documents give higher scores. Satisfactory results are obtained when the score is divided by (# of concepts per document)$^{7/8}$. Fig. 4 represents the same ADI sample as Fig. 3, except that the new scoring function h = g (# concepts per document)$^{7/8}$ is used. Unlike the function g, h seems to be independent of the number of concepts in the document.

B) Movement of Documents

The second problem is clearly indicated by examination of the results of the classification. Table 1 shows the initial and final clusters for the ADI collection. The problem occurs because the documents tend to "stick" to the clusters that they are already in. This problem is solved by a method similar to that used by Doyle.

During the first few iterations, documents should be allowed to move freely from cluster to cluster, until a <u>nucleus</u> is formed within each cluster. The nucleus consists of those documents that are most highly correlated to one another. Once the nucleus is formed, these documents will probably not move from their present clusters. Clusters can be forced to contain only very highly correlated documents by raising the cut-off value T, assuming that documents with the highest scores are most similar to the other documents in the cluster. This assumption is investigated later. However, raising the cut-off value results in a larger number of loose documents. This is resolved by repeating the

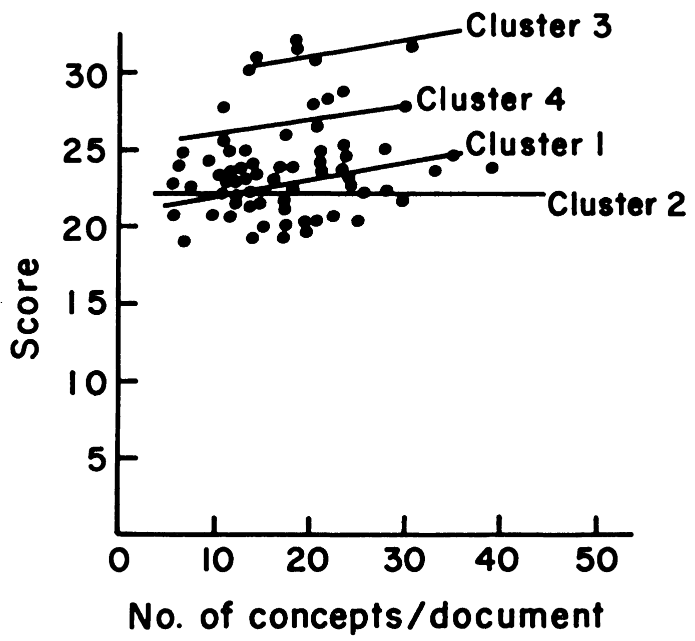

Illustration of Modified Scoring Function

Fig. 4

Cluster	Initial Documents	Final Documents
1	1 - 12	1 - 11, 13, 21, 30, 33, 34, 40, 43, 51, 68
2	13 - 24	3, 10, 13 - 24, 26, 33, 34, 53, 69, 79
3	25 - 36	9, 11, 13, 20, 22, 23, 25 - 28, 30 - 34, 36, 47, 51, 55, 65, 75
4	37 - 48	4, 7 - 9, 14, 20, 30, 37 - 48, 51, 69
5	49 - 60	1, 5, 7, 20, 30, 32, 45, 47, 51 - 53, 55 - 59, 79, 80
6	61 - 71	2, 9, 27, 30, 47, 51, 61, 62, 64 - 71
7	72 - 82	10, 40, 51, 72 - 75, 77 - 81
Loose	---	12, 29, 35, 49, 50, 54, 60, 63, 76, 82

Final Results of ADI Classification

Table 1

classification for a lower value of T, but using the clusters from the first classification as the initial clusters.

This creates the problem of how to determine the initial value of T, and how much to decrement it when the classification is repeated using as initial clusters the results of the first classification. The initial value of T should be high enough so that only those documents which score very highly against profile P_j are assigned to S_j. One method of achieving this is to pick T so that the clusters after the first iteration average q documents, where q is small compared to the total number of documents. In the experiments run so far, q is arbitrarily set at 4. After termination of the first classification, a nucleus is formed within each cluster. T is now chosen so that a certain percentage of the loose documents are assigned to clusters after the first iteration of the second classification. Assuming it is desirable to have approximately x percent of the documents loose after the final clusters are formed, two approaches are possible:

a) T is lowered far enough so that only x percent of the documents remain loose after the first iteration; thus, after termination of the second classification, the clusters represent the final results;

b) T is lowered just enough to allow a certain percentage of the loose documents to be assigned to clusters after the first iteration; thus, the classification is repeated until approximately x percent of the documents remain loose.

Experiments performed using both methods indicate that the second approach allows greater control of the loose documents, with only slightly greater execution times. After the first classification, a large proportion of the documents still remain loose. Therefore, if x is not too high, method a decreases T by a large amount. This injects many new documents into the clusters, and several iterations are necessary before termination occurs. Also, T is chosen so that the percentage of loose documents is x at the end of the first iteration, but it is impossible to know beforehand the percentage of loose documents after the final iteration. In general, the more iterations, the more the final percent varies from the percent after the first iteration. In method b, T is lowered just enough to allow a fairly small percentage (20% in the present experiments) of the loose documents to be assigned to clusters. This normally results in only a few iterations before termination occurs; therefore, the final percent of loose documents does not vary much from the percent loose after the first iteration.

The ADI collection is reclassified using the procedures described above, where it is desired that about 25% of the documents remain loose. Once again seven initial clusters are used, and the initial value of T is calculated to be 28.2 so that the clusters after the first iteration average four documents. However, in this case cluster 3 is assigned ten documents, while clusters 1,5, and 6 contain only one document. Thus, these three clusters are eliminated, and the documents within them become loose. After termination occurs, the final clusters are used as initial clusters for the next classification, where T is set to 19.1. The process is repeated again for T = 16.8, and after termination 17% of the documents remain loose. Table 2 shows the final results of this classification. Compared with Table 1, many more of the documents have moved from their initial clusters.

Cluster	Initial Documents	Final Documents
1	1 - 12	------
2	13 - 24	3, 5, 9, 10, 14 - 17, 20 - 28, 30, 34, 37, 43, 45, 48, 53, 57 - 59, 64, 68, 69, 72, 79, 80
3	25 - 36	1, 2, 5, 6, 8, 11, 13, 20, 21, 24, 27, 28, 30, 36, 39, 41, 43, 47, 51, 53, 55, 56, 58, 61, 62, 65 - 68, 70, 71 79, 80
4	37 - 48	7, 31, 42, 44, 46
5	49 - 60	------
6	61 - 71	------
7	72 - 82	4, 9, 19, 32, 40, 51, 73 - 75, 78, 81
Loose	---	12, 18, 29, 35, 38, 49, 50, 52, 54, 60, 63, 76, 77, 82

Final Results of New ADI Classification

Table 2

C) Initial Clusters

In the present study, the initial clusters are determined by assigning the first p (or possibly p+1) documents to cluster 1, the next p (p+1) to cluster 2,..., and the final p to cluster m, where p = (total number of documents) / m. Since the nucleus of each cluster depends quite strongly on the initial clusters, it is not surprising that different initial clusters lead to different results. If the initial clusters are chosen at random, it is unlikely that the documents within each cluster are very similar. Thus, the nucleus of each cluster might not be very tight.

This problem is solved by insuring that the initial clusters contain at least a few documents that are highly correlated. In the ADI and Cranfield collections, the order of the documents is such that many adjacent documents are quite similar; therefore, most of the initial clusters contain a few highly correlated documents. In collections where the order of the documents is random, a simple, fast clustering scheme can be used to determine the initial clusters. This type of an algorithm need only perform document-document correlations within a fraction of the document space, and therefore, should not take up much time.

D) Evaluation of Results

The assumption was made earlier that those documents of a cluster S_j that score highest against the corresponding profile P_j are most similar to the other documents in the cluster. The phrase "most similar" is used to mean "correlate most highly", where a standard correlation function is used. Table 3 compares the score of each document to the average correlation (unweighted cosine function) of each document with

Cluster 1			Cluster 2		
Document	Score	Avg. Corr.	Document	Score	Avg. Corr.
25	19.1	.08	8	18.7	.09
5	19.6	.12	5	18.8	.12
64	20.1	.10	20	18.8	.12
23	20.2	.13	68	18.9	.10
27	20.3	.10	2	19.2	.12
34	20.6	.11	70	19.2	.13
15	20.6	.11	39	19.2	.14
37	20.7	.14	28	19.3	.11
48	20.8	.12	58	19.4	.12
58	20.9	.12	36	19.5	.14
28	21.0	.12	61	19.6	.11
53	21.0	.14	56	19.6	.14
20	21.0	.14	66	19.7	.12
68	21.1	.12	67	19.9	.12
80	21.2	.14	80	20.0	.14
57	21.2	.13	43	20.0	.14
59	21.3	.15	33	20.0	.15
14	21.4	.13	21	20.0	.15
16	21.5	.13	11	20.0	.14
79	21.5	.15	65	20.0	.15
43	21.6	.16	27	20.0	.13
24	21.6	.15	71	20.1	.14
69	21.7	.14	41	20.2	.15
26	21.7	.16	79	20.4	.16
17	21.8	.15	24	20.4	.15
72	21.8	.17	13	20.5	.15
21	22.0	.17	51	20.6	.12
3	22.0	.17	53	20.7	.16
9	22.1	.17	6	21.0	.18
30	22.2	.17	62	21.4	.18
22	22.3	.17	55	21.5	.17
45	22.4	.18	1	21.7	.20
10	22.4	.17	30	21.8	.18

Cluster 3			Cluster 4		
Document	Score	Avg. Corr.	Document	Score	Avg. Corr.
31	31.0	.21	32	24.5	.10
46	31.2	.05	51	25.0	.15
44	31.3	.14	74	26.0	.16
7	31.8	.24	4	26.3	.18
42	33.2	.28	75	26.4	.16
			9	26.5	.19
			19	26.9	.17
			73	27.0	.19
			78	27.1	.18
			40	27.4	.24
			81	27.6	.21

Score vs. Average Correlation for ADI Classification

Table 3

every other document in the cluster. The documents are arranged in ascending order by scores, and hopefully, the correlations will also appear in ascending order. As the table indicates, there is a strong tendency for the higher scores to correspond to the higher correlations. Table 4 illustrates the same results for three out of seven final clusters from the Cranfield collection.

So far nothing has been said about how to choose the base value (section 3) that is used to compute rank values. This integer has an important effect on the type of clusters produced. Recall that the rank value of a concept equals the base value b, minus its rank. Suppose a cluster S_j contains four documents $d_1 - d_4$, and a total of twenty different concepts. The lowest possible rank value for any concept = b - 4, since 4 is the lowest possible rank. If b = 20, then the lowest rank value is 16, while if b = 5, the lowest rank value is 1. Consider a document d_i which is the same as d_1 except for one concept, and assume this concept does not occur in P_j. With b = 20, $g(d_i,P_j)$ is between 16 and 20 points less than $g(d_1,P_j)$; with b = 5, $g(d_i,P_j)$ is only 1 - 5 points less.

Since large clusters have profiles containing many concepts, the chances of a random document d_i having concepts in the profile of a large cluster are greater than the chances of d_i having concepts in the profile of a small cluster. Therefore, if b is high, d_i will score much lower against the profile of the small cluster, and large clusters will tend to capture all the remaining loose documents at the expense of the smaller clusters. Experimental results support this hypothesis, i.e., a large base value produces a few clusters with many documents, and many clusters with only a few documents.

Cluster 1			Cluster 2		
Document	Score	Avg. Corr.	Document	Score	Avg. Corr.
26	22.0	.12	38	19.9	.12
6	22.3	.13	97	20.2	.13
7	22.4	.13	15	20.3	.11
117	23.0	.14	1	20.3	.13
2	23.1	.14	34	20.4	.13
121	23.2	.15	145	20.4	.14
13	23.3	.15	171	20.8	.12
19	23.4	.16	172	20.9	.13
60	23.7	.15	30	20.9	.14
23	23.8	.17	4	21.1	.15
18	23.8	.17	140	21.1	.15
44	24.1	.18	72	21.2	.15
183	24.2	.17	138	21.3	.15
116	24.3	.18	143	21.3	.15
128	24.5	.18	141	21.3	.14
61	24.6	.18	27	21.6	.13
9	24.6	.18	36	21.7	.13
197	24.7	.19	157	21.7	.17
16	24.7	.20	59	21.8	.16
198	24.7	.17	156	21.8	.16
3	24.8	.18	200	21.9	.16
25	24.9	.20	32	22.0	.18
28	25.0	.21	137	22.4	.15
115	25.1	.21	29	22.5	.19
58	25.6	.21	148	22.5	.17
181	25.6	.20	57	22.8	.18
56	25.9	.21	128	22.8	.15
160	26.6	.23	44	23.1	.19
			31	23.2	.19
			139	23.9	.19
			56	23.4	.18
			160	24.3	.18
			58	25.3	.21

Cluster 3		
Document	Score	Avg. Corr.
179	31.4	.19
154	32.5	.24
79	32.8	.27
133	32.9	.29
134	33.4	.28
77	33.7	.27
132	33.8	.32
78	34.1	.30
76	34.3	.34
74	34.4	.34
75	34.5	.36

Score vs. Average Correlation for Cranfield Classification

Table 4

If, on the other hand, b is set so that the lowest rank value in an average cluster is 1, then there is a tendency for small clusters to get larger and large clusters to get smaller. In smaller than average clusters, all the rank values are high, since there are only a few different ranks. In larger than average clusters, the rank value as defined might become zero or even less than zero. In these cases, it is redefined to be 1, but then it is possible for many concepts to have a rank value of 1. Thus, a document often scores higher against the profiles of smaller clusters.

The results of the Cranfield classification clearly indicate the ability of a document to score higher against profiles of smaller clusters. During the classification, nine clusters are generated, and cluster 9 starts to grow much larger than average (average = 22 documents). It keeps growing until it contains 27 documents, and then it starts to oscillate. The following numbers indicate the number of documents in cluster 9 on successive iterations: 27, 21, 34, 17, 56, 0! Thus, cluster 9 is eliminated. The same thing happens to cluster 8 on the next few iterations. Although this tends to keep the size of the clusters somewhat uniform, it is not desirable to throw away a cluster which might contain many highly correlated documents. One solution which might be implemented is to split up large clusters into several smaller ones; i.e., classify the documents within a single cluster. If the number of documents in the cluster is not too large, it might be practicable to use an N^2 algorithm to do this.

7. Conclusion

The classification algorithm that has been described in sections 5 and 6 requires the following parameters as input:

a) maximum number of clusters desired;

b) approximate percentage of loose documents desired;

c) decision on whether or not loose documents should be "blended" into the nearest cluster at the end of the classification;

d) amount of overlap desired.

The first parameter specifies the number of initial clusters that are formed. If no clusters are eliminated during the evaluation, then the maximum number are actually generated. The experiments run so far indicate that the number of clusters produced is usually only about 60% of the maximum.

The next two parameters determine the "tightness" of the final clusters; the higher the percentage of loose documents, the tighter the clusters. If no loose documents are desired, parameter b can be set to 0, but very low percentages increase the running time of the program. Almost identical results are obtained in less time by specifying about 15% loose, and then asking for all loose documents to be assigned to the cluster to which they score highest.

The last parameter determines the amount of overlap. This number corresponds to a in the formula

$$T_{n-1,i} = \begin{cases} H_{n-1,i} - a \cdot (H_{n-1,i} - T), \text{ if } H_{n-1,i} > T \\ T \text{ otherwise} \end{cases}$$

which was mentioned in section 5. When a = 0, no overlap is produced, and with a = 1, the maximum amount of overlap is produced. The actual percentage of overlap for a given value of a depends on the collection, but results indicate that 10% overlap for a = .4, and about 20% for a = .6.

Although the algorithm is not guaranteed to terminate, convergence has always been obtained in practice. In order to prevent the program from looping in cases of non-convergence, the algorithm can be modified to permit a maximum of n iterations, whether or not convergence is obtained. The results indicate that clusters change very little after about four or five iterations, so that this modification would not make much difference in the final clusters.

The true evaluation of the final clusters can only be made by actually performing two-level searches on the clustered document space. However, the algorithm is sufficiently general to allow for the evaluation of many different types of clusters.

References

[1] K. S. Jones, D. Jackson, Current Approaches to Classification and Clump - Finding at the Cambridge Language Research Unit, The Computer Journal, Vol. 10, May 1967.

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

[3] R. M. Needham, The Termination of Certain Iterative Processes, The Rand Corporation memorandum RM-5188-PR, November 1966.