VIII. A SMART CLUSTERING PROGRAM

A. Priver and M. E. Lesk

1. Clustering

One of the basic features of an information retrieval system is the ability to group together, in a meaningful manner, terms and expressions in the documents in the collection under consideration. Such grouping, or clustering, of terms enables the contents of any document to be more accurately classified and described. For example, if the terms "missile" and "submarine" were frequently found together in a document, then one of the word clusters for this document would be "submarine missile." Hence, when a request is fed into the system, the process of retrieving information becomes simpler and more reliable.

In the SMART system each word in a document is not considered in its own right, but is assigned various numbers, called concept numbers, which describe the word. In this way each word is put into one or more categories or classes, much as in Roget's Thesaurus, for example. Thus, when clusters are sought in documents it is not the words themselves but actually the corresponding concept numbers which are clustered.

A clarification of the above is best given by considering the following formulation of the "clustering problem:" given a set of documents and a mapping of the index terms (words) into objects (concepts), to find groups of objects which "look alike," but which do not resemble any objects outside of the group. This rather weak, intuitive statement is sufficient to initially describe the situation because some basic notions in the area of clustering have not yet been formalized.

However, more precision can be introduced in the context of the SMART System. Bearing this thought in mind, some notions will now be discussed which will help to sharpen the definition of clustering. The first such idea to be presented is that of the "connection" between two terms. This notion describes the frequency of co-occurrence of the terms in the document collection, or within the individual documents. Basically, an attempt is made to state how closely two terms are related to each other.

The most common procedure for arriving at this connection is to first form what is called a concept-sentence, or concept-document, matrix. There is one row for each concept number and one column for each sentence (or document). Any element a(i,j) indicates the number of times that concept i appears in sentence (or document) j.

After forming this matrix, it is then possible to define some measure of the correlation between rows or columns. The one chosen here is the method known as the cosine correlation:[6]

$$c(x,y) = \sum_{2=1}^{n} x(i)*y(i) \bigg/ \sqrt{\sum_{2=1}^{n} x(i)**2 \sum_{2=1}^{n} y(i)**2}$$

The "connection" between two terms will be taken to mean the correlation between them.

After establishing the notion of the connection between a pair of elements, an improvement can be made in the definition of cluster. A cluster may now be considered to be a subset of objects, A, which have a greater number of connections to each other than to objects in the complement of A (defined as B), with the reverse holding for objects in B; that is, elements of B have more connections to each other than to elements of A. A similar definition is used by Dale, Dale, and Pendergraft,[1] who in turn attribute it to Needham.

Note that a cluster is only defined relative to the connection definition being employed. Hence, various correlations, aside from the cosine one above, could give different results. Further, if the concept-sentence matrix is binary (in which case each entry simply indicates whether or not the concept appears in the sentence and not how many times it does so) then many specialized correlations may additionally be used.

Another important problem affecting the clustering problem is that of choosing the initial concepts in the system which will be used to form the clusters. This section is not concerned with this problem, but assumes that the concepts chosen for the SMART system are adequate for the task of providing good clusters. Furthermore, it is perfectly reasonable to have any concept appear in either more than one cluster or in no clusters. In fact, those concepts which do not appear "often enough" in the document under consideration may not be included in the clustering process.

Clusters are thus not necessarily and not generally groups of synonyms, but simply those terms which tend to co-occur.

The techniques employed in the clustering program about to be described are a combination of one described by Bonner,[4] and one in the papers by Dale, Dale, and Pendergraft.[1,2,3]

The initial step in the clustering process consists of forming the concept-concept, or similarity, matrix. The entry in row i and column j of this matrix is a measure of the connection between the concepts i and j, with the cosine correlation being used. The similarity matrix may be either binary or numeric, independently of the state of the original concept-sentence matrix. Since it is generally the case that two concepts which have high correlations to the same terms are also closely related to each other, Bonner proposes that the similarity matrix of this similarity matrix be formed, using one of the connection relations. Inasmuch as the result of this operation is again a similarity matrix, the procedure may be repeated as many times as desired. Bonner claims that this process is carried out in order to better define clusters with loose internal connections, and to allow better separation of overlapping clusters.

Having thus formed a similarity matrix, Dale's procedure is then used to find clusters. The following notation may be used:

U:	the set of all the n concepts
A:	a subset of U with elements a(i), i=1,t
B:	the complement of A with elements b(i), i=1,r
x:	a concept in U (r+t=n)
c(x,y):	the element of the similarity matrix denoting the connection of the pair of concepts x and y. (Assume that c(x,x) = 0).

C(x,A): $\sum_{2=1}^{t} c(x,a(i))$ (the total connections of any concept x to all concepts in A)

C(x,B): $\sum_{2=1}^{r} c(x,b(i))$

b(x,A): C(x,A) - C(x,B). The <u>bias</u> of a concept x to set A is the number of connections to A minus the number of connections to B. This number may be either positive or negative.

A subset A is a cluster if $A = \{x : b(x,A) \geq 0$ and $b(y,A) < 0$, for all y in $B\}$: All members of A have positive or zero bias to A, and members of B have negative bias to A.

In order to get good clusters, it is necessary to have good initial partitions. Many methods exist for finding initial partitions but none of them can guarantee the goodness of the final clusters. Dale suggests that one arbitrarily consider n initial partitions of U, with partition j containing concept j and all concepts with a nonzero connection to j. The following series of operations is to be performed on each initial partition:

(1) Compute b(x,A) for all x in U.

(2) If $b(x,A) \geq 0$, transfer x to A, if not there already.

(3) If $b(x,A) \leq 0$, transfer x to B, if not there already.

(4) Recompute the biases after each transfer.

(5) Repeat steps 1-4 until no transfers are made for one complete scan.

(6) A is a cluster.

For the present no consideration is given to the cases in which A is either the empty set or the whole universe. Instead, only relatively small clusters are of interest. Since the clusters produced by the above procedure may be large, provision must be made for reducing their size until each cluster has fewer than some maximum number of elements in it.

One possible method for performing this reduction is to remove a concept from A which has a bias less than some value b', and then to recompute the biases of each remaining concept over U. This transferring is done until all the remaining elements have a bias to the reduced set A' greater than b'. If A' still has too many elements the process may be repeated with a higher b'. Thus each cluster may have a different threshold level because b' always starts at the same initial value and is increased in equal steps until the final cluster is found. If the final cluster has no elements, then the process is repeated from the beginning with the step length for incrementing b' cut in half.

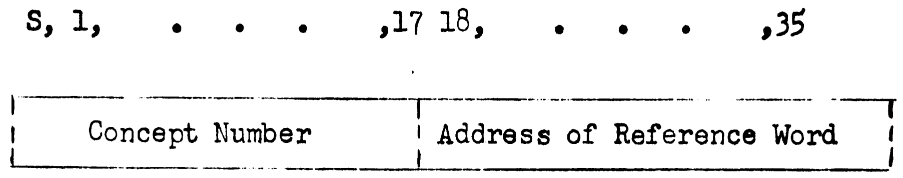
The procedures outlined above produce clusters, but not necessarily all clusters. Part of the difficulty lies in finding good initial partitions. A further problem is the choosing of a good bias level b' to get clusters which are small, but which contain more than one element. A computer program has been written to implement the above heuristic methods and is described in the following part.

2. The Program

The main program first calls the subroutine SREAD to read in the first one thousand words of a document. The next program is TREADL, which reads in the thesaurus, either regular or null thesaurus. The thesaurus is actually a table giving the concept numbers corresponding to each word which may occur in a document. The null thesaurus assigns a unique con-concept number to each word. A call to SEGMNT creates a 10,000 word array in which each word read is represented by a ten-word vector. The first word stores the sentence number in the decrement and the number of the word in the sentence in the address portion. Words 2-5 contain the English text (a word consists of no more than 24 characters). Words 6-8 contain the concept numbers corresponding to the original word, and words 9-10 are of no interest to the clustering program. Distinct concept numbers are stored in the decrements and in the address portions of words 6-8, so that any word may map into as many as six concept numbers. A concept number of zero indicates that a word is not found in the thesaurus. The zero also acts as a code indicating the end of the list of concept numbers for those words with fewer than six associated concept numbers. A concept number greater than 1000 identifies the word as a common one; such a word is then skipped over by the clustering program. A common word has only one concept number associated with it. The routine that actually maps each word into the appropriate concept numbers is called LOOK. The above procedure is currently incorporated in the SMART system.

Subroutine CLSTR controls the entire process of converting the 10,000 word array into a concept-concept matrix. It first uses SKWZ to remove words 2-5 and 9-10 from the array, thus reducing it to 4000 words. The rest of this array is zeroed out in order to make available additional storage space in the machine. After this routine SPLIT is used. It splits up the condensed array into 1000 groups, each of which contains one word of the text and its associated concept numbers, and then calls UNPACK to form the concept-sentence matrix. UNPACK operates on one group at a time.

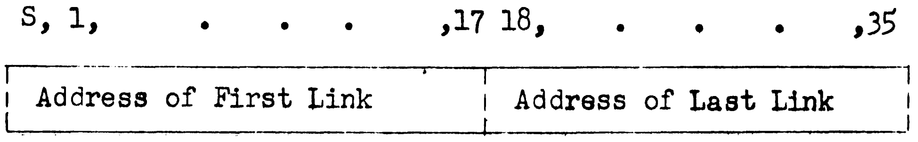
It takes the concept numbers which correspond to each word in the document and determines, for each concept, by a call to LKUP, whether this is its first occurrence in the document. If it is, INSERT is used to place the concept in the numerically ordered table of concept numbers. Each word in this table is divided into two parts: The decrement contains the concept number, and the address has a pointer indicating the reference word for this concept number.



Entry in the Table of Concept Numbers

Figure 1.

The reference word is used for finding either the beginning or the end of the chain for the desired concept number. This chain has one entry for each sentence in which the concept appears. Each link is described below in detail. The decrement has the address of the first link in the chain, while the address portion contains the address of the last link. This word is used for adding something on to the end of the chain since it eliminates the necessity of going through link by link to find the end.



Reference Word

Figure 2

Each word in the chained list is broken up into three parts.

S,1...,8	9, . . . ,20	21, . . . ,35
Weight	Sentence Number	Address Next Word

Entry in the Chained List

Figure 3

As indicated in Fig. 3 the first part of the word is taken to be positions S, 1-8, which contain the weight assigned (the value of the coefficient in the concept-sentence matrix indicating how many times the concept appears in the sentence). Positions 9-20 represent the sentence number (or column number) in the array corresponding to this term. Positions 21-35 contain the address of the next word in storage pertaining to the same concept number. A zero address indicates the end of the chain. The above decomposition restricts the weight of any term to be less than 512, and the number of sentences in the document to be less than 4096.

After finding the concept in the list of concept numbers the last link in the chain is found. If the sentence number of this link is the same as the current sentence, simply add 1 to the weight of the word in the chain. If the current sentence number differs from that of the last word in the chain, then this indicates that the concept is appearing in this sentence for the first time. A new word must be added to the end of the chain and the addresses in both the reference word and the former end word must be changed so as to point at the newly added link.

After thus forming the concept-sentence matrix, subroutine CONCON is called upon to form the concept-concept matrix. This matrix is stored in the same way as the concept-sentence matrix. There is a list of concept numbers and a chained list for each one of the concept numbers. The only differences are in the interpretation of positions 9-20 of each link, which now indicate a concept number instead of a sentence number, and in the meaning of positions S, 1-8, which now contain, if this entry is the one for concept B in the chain for concept A, $\sum A(i)*B(i)$. ($A(i)$ indicates the number of times that concept A appears in sentence i.) This product is found by use of CORRL, which takes the chained lists for the two concepts A and B (that is, it takes the rows in the concept-sentence matrix corresponding to A and B) and sums the products of those entries which correspond to the same sentence numbers. By virtue of the symmetry of the concept-concept matrix only half of it need be stored.

Next the clustering procedure is executed using subroutine QDALE. Bonner's iteration scheme is applied to the concept-concept matrix as many times as is desired. Then the first initial partition is formed by choosing the first concept and finding all those concepts with a nonzero correlation to it. SELECT is used for carrying information to QDALE from the concept-concept matrix. For each concept number, SELECT goes through the chained list one link at a time and gives to QDALE the corresponding column number and correlation. After forming the partition, Dale's method is applied. Concepts are shifted until a cluster is found. If the cluster is too big (that is, more than six concept numbers in this case) the threshold bias is raised and the procedure is repeated. The final cluster is then printed out, and the next initial partition is considered.

The cluster formation procedure begins with one concept number which is used to form the "initial partition." QDALE applies the method of Dale and Dale[2,4] to this initial partition. Each concept number is used in turn to form an initial partition which consists of all concepts having a nonzero correlation with this concept. This cluster is now adjusted by computing the bias of each concept. The correlations generated by CORREL are summed for the member and nonmember concepts. The array FACTOR is used to keep track of the cluster numbers. If concept I is in the cluster, FACTOR(I)=+1; if concept I is not in the cluster, FACTOR(I) =-1. The bias is compared with the threshold THRESH, and if it is greater than THRESH the concept should be in the cluster. If it was not there originally, it is placed there. Similarly, concepts in the cluster whose biases are less than THRESH are removed from the cluster if they were there originally. After all concept biases have been examined, the program checks to see if any concepts were moved in this iteration. If so, the process is repeated. When the cluster has converged, its size is compared with an arbitrary size limit IMAX, and if it contains more than IMAX concepts the iterations are repeated with a larger THRESH.

The program is debugged and some trial runs are being performed on actual data in order to determine the efficacy of the methods described above. Some threshold parameters are still being experimentally determined. Also, Needham has recently suggested some possible improvements in Dale's methods which may give better results, and which will be included in a revised version of the program. In one test case which has been run both the clusters in the sample set of data were found by the program. One cluster had four elements, the other six, and there were three elements common to the two clusters. Thus far the results are quite encouraging.

3. Example

As an illustration of the procedures described above, consider the following example. The data were read in and the following concept-sentence matrix was produced.

		Sentences					
		1	2	3	4	5	6
	1	1	0	0	1	1	1
	2	1	0	1	0	0	0
	3	1	1	0	1	2	1
	4	1	0	1	1	2	3
	5	0	1	0	0	0	0
Concepts	6	1	0	0	0	3	1
	7	1	0	0	0	1	1
	8	1	0	0	0	2	2
	9	0	0	0	0	2	0
	11	0	0	0	0	2	3
	13	0	0	0	0	2	0
	14	0	0	0	0	1	0

A total of twelve different concepts occurred in the six sentences. The first initial partition consisted of all those concepts with a positive connection to concept number one. This partition included all the concepts except the fifth one. The size of the cluster was gradually reduced until at the b' cutoff value of 6.29, a six element cluster was produced containing concepts 3, 4, 6, 7, 8, and 11. The second initial partition was based upon connections to the second concept. It consisted initially of concepts 1,2,3,4, 6,7, and 8, and resulted (with b' of 2.43) in the four-element final cluster composed of concepts 1, 4, 7, and 11. The second case shows that it is possible for a concept (No. 11 here) to appear in the final cluster and not in the initial cluster. All the remaining initial partitions produced the same clusters already found by the first two partitions.

References

1. A. G. Dale, N. Dale, and E. D. Pendergraft, "A Programming System for Automatic Classification with Applications in Linguistic and Information Retrieval Research," International Study Conference on Classification Research, Elsinore (1964).

2. A. G. Dale and N. Dale, "Some Clumping Experiments for Information Retrieval," Linguistic Research Center, Austin, February (1964).

3. A. G. Dale and N. Dale, "Clumping Techniques and Associative Retrieval," NBS/ADI Symposium on Statistical Association Methods for Mechanized Documentation, Washington, March (1964).

4. R. E. Bonner, "On Some Clustering Techniques," *IBM Journal of Research and Development*, Vol. 8, No. 1, pp. 22-32, Yorktown Heights, January (1964).

5. *Information Storage and Retrieval*, Report No. ISR-7, Report to the National Science Foundation, Cambridge (1964).

6. *Information Storage and Retrieval*, Report No. ISR-8, Report to the National Science Foundation, Cambridge (1964).

7. R. M. Needham and K. Sparck Jones, "Keywords and Clumps," *Journal of Documentation*, Vol. 20, No. 1, March (1964).

8. R. M. Needham, "A Method for Using Computers in Information Classification," IFIPC, Munich (1962).