Incorporation of Relevance Data in the Term Discrimination Value

Tesfaye Biru

A study submitted in partial fulfilment of the requirements for the degree of Master of Science in Information Studies

at

The University of Sheffield

SEPTEMBER 1987

Incorporation of Relevance Data in the Term Discrimination Value

Tesfaye Biru

A study submitted in partial fulfilment of the requirements for the degree of Master of Science in Information Studies

at

The University of Sheffield

SEPTEMBER 1987

ACKNOWLEDGEMENTS

I would like to thank Dr. Peter Willett for being an ever-ready and stimulating supervisor, and for his overall patient assistance during the project; also Mr A. El-Hamdouchi for developing part of the program on which this work is based. I am grateful to the staff of the Sheffield University Computer Center for their help and co-operation; to UNESCO for financing my study.

Contents

ABSTRACT

Indexing in information retrieval is used to obtain a suitable vocabulary of index terms and optimum assignment of these terms to documents for increasing the effectiveness and efficiency of the the retrieval system. A great many automatic indexing models have been developed over the years in an effort to produce indexing methods that are both effective and usable in practice. One of the most elegant approaches for automatic selection and weighting of index terms is the term discrimination value that has been developed by Salton and his co-workers. This model ranks the index terms in accordance with how well they are able to discriminate the documents of a collection from each other; that is, the value of an index term depends on how much the average separation between individual documents changes when the given term is assigned for content identification. It is suggested that the most useful index terms, those which achieve greatest separation, are the medium frequency terms.

Since the basic requirement in effective retrieval is the separation between documents which are relevant to a given query and documents which are not relevant to that query, a more complete picture of a term behavior may be obtained by the consideration of its ability to effect greater separation between relevant and non-relevant documents while at the same time moving relevant documents close to each other.

This study was aimed at testing the extent to which the discrimination value model considers relevance characteristics of documents in ranking the index terms. An over-view of the more important ideas current in automatic indexing is provided. The term discrimination value model is discussed in greater detail. An efficient technique for computing exact term discrimination values for relevant - non-relevant document distinction is introduced. The study is conducted using the KEEN, CRANFIELD, EVANS, HARDING and LISA document collections and their associated queries and relevance judgements.

While some of the results are consistent with those derived by previous workers, in some cases, specially in the case of relevant - relevant discrimination, the results obtained appear to be in complete disagreement with that of Salton's theory: that the medium frequency terms are not the most useful terms.

Chapter 1

Introduction

The task of document indexing involves the description of the contents of each of the documents in databases by means of a series of identifiers or index terms, capable of representing the contents of the document, and adequate for the retrieval of the document in response to subsequent queries. Traditionally this has been performed by manually trained indexers, who are knowledgeable about the subject matter of the database, through scanning of the entire text or selected portions of the text, like titles, abstracts or topic sentences. However, while manual indexing can give good results in principle, a number of experimental studies (see for example, Salton, 1971, Salton, 1975b) indicate that it is fraught with difficulties. One obvious limitation is that a number of possible subject interpretations can be given to the same document by different indexers at different times and different clients.

In order to obviate the difficulties encountered with manual indexing it has been suggested that it should be possible to introduce automatic indexing techniques, in which the task of indexing is carried out by automatic, rather than manual, means. Substantial evidence now indicates that simple automatic indexing methods are fast and inexpensive as compared to manual indexing (Salton and McGill, 1983).

Much early research in automatic indexing focused upon the use of linguistic, rather than statistical, techniques to identify terms and phrases in natural language texts. Such methods are variously described as linguistic analysis, non-statistical methods of automatic indexing, and automatic syntax analysis. However, these attempts have met with little success (Sparck Jones, 1982, Salton, 1975b). Generally speaking, the current literature on automatic indexing theory emphasizes the statistical analysis of document and query texts. To this end, a wide range of automatic indexing techniques have been suggested. Although these methods do vary, there seems a general agreement that an automatic indexing system should consist of a **term selection** module, which is responsible for the selection of the indexing terms, a **conflation procedure** , which is used to ensure that different forms of a given word are recognized as being equivalent, and a **weighting mechanism** which assigns measures of relative importance to the selected terms.

Most automatic indexing efforts start with the observation that the frequency of occurrences of individual word types in a database has something to do with the importance of these words for purpose of content representation, a technique pioneered by Luhn (1957). According to Luhn's observation, a term which occurs very frequently in a database is unlikely to be able to discriminate sufficiently between relevant and non-relevant documents if it is specified in a query; a very infrequently occurring term, conversely is well able to discriminate but, by its very nature, is most unlikely to be specified in a query. Therefore, the most useful terms for retrieval purposes are those of intermediate frequencies of occurrence. Thus, terms representing the content of a document can be obtained simply by counting the collection frequency of each term, i.e., its frequency of occurrence in the database; and then using as indexing terms for each document those terms contained in it which have intermediate frequencies of occurrence.

Another model which provides an interesting theoretical basis for the use

of medium frequency terms is the term discrimination model advocated by Salton (1975a). Salton and his co-workers have experimented with several approaches to computing the significance of terms and their usefulness as discriminators. They assume that a good index term is one which, when assigned as an index term to a collection of documents, can best separate the documents (as dissimilar as possible), whereas a bad term is one which moves the documents closer to each other (more similar). This term significance is quantified through a term discrimination value which for a particular term measures the increase or decrease in the average inter-document similarities. Therefore a good term is one which, on removal from the collection of documents leads to an increase in the average similarity, whereas a bad term is one which, on removal, leads to a decrease. Several studies (Salton et al, 1975b, Salton and Wu, 1981) have demonstrated a strong relationship between term discrimination value and collection frequencies, with the most highly discriminating terms being those of intermediate frequencies of occurrence in document collection.

There are a number of discussions in print now which cover the use and calculation of term discrimination values. One obvious limitation of the model is that the calculation of the discrimination values involves extensive computation. A number of workers (Salton et al, 1975a, Crawford, 1975) have suggested ways of reducing the computation involved at the expense of approximating the actual values. However, more recent work (El-Hamdouchi and Willett, 1988a) has suggested an efficient algorithm for the calculation of exact term discrimination values that may be used when the inter-document similarity measure used is the cosine coefficient and when the document representatives have been weighted using one particular weighting scheme. The basic assumption in the term discrimination theory is that a greater separation between documents will enhance retrieval effectiveness but that less separation will depress the retrieval. Although this is reasonable, what is really required is that the relevant documents become more separated from the non-relevant ones. However, despite extensive research in the field,

little has been said about the effect of the term discrimination model on the separation of relevant - relevant and relevant - non-relevant documents. In fact, in most cases it has been considered convenient to assume that good discriminators separate relevant from relevant less while they separate relevant from non-relevant more. However, there does not appear much reasonable justification for this assumption.

The aim of this study is, then, to test the extent to which the term discrimination model considers relevance characteristics of documents in computing the discriminating power of an index term, a situation which will make the retrieval of relevant documents more likely and the retrieval of non-relevant document less likely. To this end, basic techniques for choosing indexing terms, conflation procedures and assigning weights to the terms are examined in the next chapter. Chapter three concentrates on the theoretical formulation of the term discrimination model. Also covered are the various procedures suggested for calculating term discrimination values and the comparison with other term weighting schemes. The fourth chapter is devoted to an account of the actual experiments carried out and the description of the data sets used. The discussion of results obtained and the conclusions reached are presented in chapter five.

Chapter 2

Automatic Indexing

2.1 General Consideration

Under the general heading of automatic indexing, a whole range of approaches and specific procedures have been advocated in the past. This is an attempt to cover briefly the main options in a somewhat systematic way. Before describing the various methods that have been suggested, the following few conventions are in order.

The term 'automatic indexing' is used here to refer to a process in which documents and queries are analysed and described automatically using an automatically created index language for retrieval purposes. Areas of inquiry which are closely related to automatic indexing include automatic abstracting and automatic classification. These topics will not be explicitly dealt with, although some common points are evident. Another limitation of the study is that only statistical approaches to automatic indexing will be treated. Syntactic and semantic theories of automatic indexing are particularly excluded from consideration, a good account of such approaches may be found in Sparck Jones and Kay (1973); a useful general survey of these methods has been presented by Montgomery (1972).

The statistically based techniques for automatic indexing do vary, but they are all based on the idea that the frequency of occurrence of an index term (a word in a document) is a useful indication of its relative importance in describing a document. This idea is in part based on Zipf's law, which states that, when the distinct words in a body of text are arranged in decreasing order of their frequency of occurrence (most frequent words first), the frequency of a given word multiplied by the rank order of that word will be approximately equal to the frequency of another word multiplied by its rank (Booth, 1967)

$$frequency \times rank \simeq constant$$

This idea is used as the starting point in a number of methods for determining term significance that will be discussed in the subsequent sections of this chapter. To render this discussion meaningful, automatic indexing is conveniently viewed as a three stage process : (1) the identification of keywords descriptive of individual documents or queries, (2) detecting equivalent forms of selected terms, and (3) weighting (i.e., assignment of relative importance value for each term).

2.2 Term Selection

The task of identifying a set of terms or keywords, from the full-text of a document or a query statement, that are capable of representing the content the given document is a necessary step in the development of a fully automated indexing and retrieval system. Historically, the selection of such terms involves either intellectual or automatic processing and is either statistically or linguistically based. Early approaches towards the automatic selection of indexing terms were aimed at emulating the intellectual process automatically. However, the results obtained from such approaches were found to be generally unsatisfactory. Thus the great bulk of automatic indexing research has involved the use of statistically based, rather than linguistically based,

techniques. The focus in this section, therefore, is on the extraction of possible index terms from documents or document excerpts using statistically based techniques.

The most common form of automatic selection of index terms is by extracting terms from the complete document or query text, and this technique was pioneered by Luhn in the late fifties. Using the Zipf's law above as a starting point, Luhn (1957) carried out extended studies on automatic indexing. He observed that a high frequency term, i.e., a term that occurs in many documents, is unlikely to be able to discriminate sufficiently between relevant and non-relevant documents if it is specified in a query; a low frequency term, on the other hand, is well able to discriminate but, by its very nature, is most unlikely to be specified in a query. Therefore, the most useful terms for retrieval purposes are those of intermediate frequencies of occurrence. This then suggests that one way of selecting terms representing the content of a document or a query is simply by counting the collection frequency of each term, i.e. its frequency of occurrence in the collection, and then using as indexing terms for each document or query those terms contained in it which have intermediate frequencies of occurrence.

The practical implementation of this procedure, however, has identified some problems. One major problem is the necessity to choose appropriate tresolds (i.e. the two values which are used to eliminate the rare terms) in order to distinguish the useful medium-frequency terms. Normally the upper treshold is not determined statistically; instead, a **negative word dictionary** is compiled, and stop words that appear in the dictionary are removed from the text. However, it has been suggested (Salton et al, 1975b) that the simple elimination of high and low frequency terms is likely to lead to recall and retrieval losses respectively (retrieval performance is often measured by parameters such as **recall**, i.e. *the ratio of relevant items actually retrieved*, and **precision**, i.e. *the ratio of retrieved items actually relevant*). In particular, the elimination of high frequency terms might produce losses in recall,

because the use of broad, high frequency terms for content identification is effective in retrieving large number of relevant items. They may also be important factors in particular situations, as in the case with legal and newspaper databases where some phrases are common. Similarly, although low frequency terms contribute little to the retrieval process (because few matches would occur on rare terms), it can be argued that when they occur they can be taken as good indicators of document relevance (Sparck Jones, 1973, Salton, 1975b). Thus the elimination of low frequency terms may produce losses in precision.

Another problem arises from the definition of the frequency, i.e., the use of absolute frequency measures for the identification of content indicators. There is some evidence (Sparck Jones, 1972, Salton and McGill, 1983) that simply counting the number of occurrences of terms in a text is not good enough because it does not take into account the number of different terms in a document. Nevertheless, Luhn's conjecture proved to be reasonably perceptive, as will be seen below.

Another approach to selecting of indexing terms is the use of the so-called term discrimination model, a theory developed by Salton and his co-workers. This theory is based on the notion of a 'space' of many dimensions, where each dimension is associated with an index term. Each document occupies a position in the space. Terms are selected according to their contribution to the overall spread of documents in the space defined by the terms. The discriminatory ability of each term can be computed as a difference in the average similarities over all document pairs in the collection before and after assignment of the term. The greater the difference in the averages, the more the space will have spread out after the assignment of a given term and the better that term will function as a discriminator. Therefore, the best discriminators for a given collection will have positive discrimination values, while worst discriminators will have negative discrimination values; while indifferent terms end up in zero values.

Salton and his co-workers carried out extended studies based on this idea. Salton and Wong (1976) noted that the best discriminators exhibit medium collection frequency; high frequency terms have negative discrimination values, while the discrimination value for low frequency terms is generally close to zero. Hence, they suggest that those terms which have a medium valued overall frequency and a skewed document distribution (i.e. they occur frequently in some documents and infrequently in others) are most useful for retrieval in indicating the content of the documents; this finding is in agreement with Luhn's wholly pragmatic indexing strategy. However, Willett (1985) noted that this identification of medium frequency terms as the most useful ones may depend upon the precise way in which the inter-document separation is calculated (as will be discussed further at a latter stage).

In practice, one possible technique for selecting indexing terms proposed by such indexing theory involves the simple elimination of bad discriminators (i.e. terms with low and high document frequency) from the vocabulary. Good accounts of such deletion procedures are presented by Salton (1975a) and Salton and McGill (1983), and recently, using a slightly different approach, by Can and Ozkarahan (1987). This approach, eliminating bad discriminators, is particularly attractive in that it significantly reduces the computer storage requirement. In addition, the calculation of the discrimination values will also be improved as less terms will be considered. However, it is quite possible that some good discriminators may appear in the sets of terms with high or low frequencies and hence will be deleted, thereby affecting the retrieval performance. This question has been addressed by Salton et al (1975b) who carried out a series of tests on different collections to study the frequency characteristics of good discriminators. They noted that the number of discriminators included in the sets of low and high frequency terms, for the collections tested, are proportionally very few. When the terms are ranked in order of increasing document frequency, they conclude that very few good discriminators are included among the bottom 70as a basis for developing an alternative strategy for selecting indexing terms,

which involves the transformation of bad discriminators into better ones.

In fact, the bulk of the discrimination value theory is concerned with methods of modifying the frequency characteristics of high and low frequency terms, thus improving the discrimination power. Salton and his co-workers argue that high frequency terms which are too general for effective use may be combined with adjoint terms as phrases to increase their discrimination power. Such a method involves the linking of high frequency keywords which co-occur in a document, resulting in a lowering of their occurrence frequencies and thereby improving their discrimination power. For instance, a phrase such as ' *programming language* ' exhibits a lower assignment frequency than either of the high frequency components ' *language* ' or ' *program* ' . Low frequency terms, on the other hand, may be transformed into higher frequency units by combining a number of related indexing terms into a term class and assigning this class as an indexing term instead of the terms within the class. Such a class will normally have a higher frequency of occurrence than any of its components and will therefore have a correspondingly higher discrimination value. An excellent discussion of the generation of indexing phrases and term classes is provided by Salton and McGill (1983).

In his review of statistical approaches to automatic indexing, Harter (1978) discusses a number of criteria for automatic selection of terms. Salton (1975a) reviews a range of frequency measures which have been used to evaluate the worth of keywords as index terms. However, despite some two decade of research, there is still disagreement as to exactly which is the most appropriate means of selecting terms (Salton, 1986). In fact, rather than applying sophisticated selection criteria, the tendency is increasingly to use all of the terms from document or query text, and then to differentiate between them by means of an appropriate weighting scheme.

Thus, the idea of selecting some of the keywords has been replaced by the simple extraction of all of them, with the exception of some of the very

high frequency terms which are eliminated by means of stop word list (words such as AND, OF, FOR etc..). All remaining words are assumed to be equally useful for indexing purposes. High frequency terms from the remainder list may be treated with the methods of adjusting frequencies of occurrence described above to increase system precision, whereas low frequency terms can be combined into classes. Term classes are often defined by a **thesaurus**, and a given thesaurus class normally includes terms that are sufficiently similar in meaning, or context, to make it reasonable to ignore their differences for indexing purposes. A great many thesaurus construction procedures have been described in the literature including manual term grouping and fully automatic methods (such as term clustering and associative indexing) (Salton, 1976, Salton and McGill, 1983). However, the use of automatically created thesauri has not proved to be very successful in practice (Sparck Jones, 1971). An alternative means of enhancing recall is by the use of conflation techniques as described below.

2.3 Conflation Techniques

Once the set of terms representing a query or document has been identified, some means must be found for overcoming the variants in word forms which are likely to be encountered in free-text systems. These variations may arise from a range of causes including the requirements of grammar, e.g. COMPUTING and COMPUTATIONAL, valid alternative spellings, RECOGNIZE and RECOGNISE, antonyms, e.g. ABILITY and DISABILITY, and problems arising from misspellings, translation, e.g. TCHEBYSHEFF and CHEBYSHEFF, and abbreviation, APPROX and APPROXN (Freund and Willett, 1982). Such equivalent word forms may be detected and reduced to a single form for the purpose of retrieval, thereby improving system performance through improving recall.

In conventional information retrieval systems, truncation, mostly right

hand truncation, is a familiar technique employed by online searchers to effect comprehensive retrieval of word variants. Truncation, however, generally results in the retrieval of some words whose roots are identical with desired terms but which are not in fact logical or semantically related to them. Over truncation occurs when too short a stem remains after truncation and may result in totally unrelated words being truncated to the same stem, as with both MEDICAL and MEDIA being retrieved by the same stem MED*; under-truncation, on the other hand, arises if too short a suffix is removed and may result in related terms being described by different stems, as with COMPUTERS being truncated to COMPUTER, rather than COMPUT* (which would also include words such as COMPUTING and COMPUTATIONAL).

One way to alleviate this problem is by the use of a **conflation procedure**, that is, mapping term variants to a single 'proper' form, usually a unique well formed nominal root for each word (Porter, 1980, Ulmschneider and Doszkocs, 1983). The most common conflation procedure is a **stemming algorithm** which reduces all words with the same root to a single stem by stripping each word of its derivatives and inflectional affixes. There are many different types of stemming algorithms which have been reported in the literature. The nature of these algorithms may vary considerably depending on whether a stem dictionary is being used, whether a suffix list is being used, but they are all based on certain principles (Lovins, 1968, Lennon et al, 1981, Ulmschneider and Doszkocs, 1983) which are briefly described below.

A stemming algorithm can be **iterative** in character, use the principle of **longest-match** assignment, or a combination of the two. An iterative stemming algorithm is based on the fact that suffixes are attached to stems one after the other. Thus as its name implies, such an algorithm involves a recursive procedure which removes the suffixes one at a time, starting at the end of a word and working towards its beginning. For instance, a word

such as WILLINGNESS might have -NESS removed in the first iteration and -ING in the second. Such an algorithm requires the construction of a dictionary which contains a list of all possible word endings. Longest-match algorithms, on the other hand, involve only a single iteration in which case, if more than one suffix matches the end of a word, the longest one is removed. This requires, however, the compilation of all possible combinations of suffixes. In order to reduce programming complexities, this list of suffixes is sorted in decreasing order of suffix length. The procedure is then to scan the suffixes in order of decreasing length. That is, the longer endings are first scanned, and if a match is not found, then the shorter ones are scanned. Thus the word RELATIVISTIC would be stemmed to RELATIV if both -ISTIC and -IC were included in the suffix listing being used. Longest-match algorithms are often easier to program but require a much longer dictionary since frequent combinations of short suffixes must be included.

Another basic characteristic of a stemming algorithm is whether it is **context-free** or **context sensitive**, where context refers to any attribute of the remaining stem. In a context-free algorithm, no restriction (except for some extreme cases such as the length of the remaining stem must not be zero, e.g. -ABILITY from ABILITY) is placed on the removal of a suffix and thus any ending which matches is accepted for stripping. Context-free algorithms are much simpler to develop and may also be more efficient at run time since no character matching need be carried out to determine the context.

In context sensitive stemming algorithms, conversely, various restrictions are placed on the usage of the suffix. Therefore, algorithms in this category require the construction of a suffix dictionary and the formulation of a corpus of rules defining the morphological context of the suffixes (Lovins, 1968). The dictionary gives the exact suffix form, while the rules define, for example, the treatment of dictionary suffix when preceded by a double consonant (such as STEMMING $\longrightarrow$ STEM), or the minimal root size that

must be retained (such as the removal of UAL from FACTUAL but not from EQUAL), and some general rules like, for example, do not remove a suffix that begins with -EN following -E, (as in SEEN). When a character string in a dictionary is encountered as a terminal string in a word, a suffix is tentatively identified. If the set of rules defining the correct morphological context for the suffix is satisfied, it is replaced by another string, either the null string (if the suffix is to be removed) or a specified replacement string (for example, to create nominal forms from adjectival forms). Both dictionary and rules require careful analysis of vocabulary and language behavior, and are thus time-consuming to create. However, generally such techniques are rewarded by high accuracy and speed, and simplicity in implementation (Lovins, 1968, Ulmschneider and Doszkocs, 1983).

In addition, stemming algorithms may also include some recoding rules which are applied after stemming has taken place. For example, the elimination of the doubling terminal consonants which occur when the word FORGETTING is stemmed to FORGETT by removal of the suffix -ING. Other recoding rules may be used to conflate such word pairs as ABSORB and ABSORPTON, ANGLE and ANGULAR.

Examples of typical stemming algorithms may be found in Lovins (1968), Porter (1980), and Ulmschneider and Doszkocs (1983). A comparative survey of wide range of conventional stemming algorithms has been carried out by Lennon et al (1981). This survey reported that relatively little difference in terms of the dictionary compression and retrieval effectiveness, between the various algorithms that were tested, despite the different means by which the various algorithms had been developed.

Although word stemming is easy to implement and provides a highly effective means of conflating words with different suffixes, there are many other types of word variant which are likely to occur in free-text databases. Attempts have been made to provide conflation mechanisms for such cases. One such technique is prefix processing (for terms such as UNIFY and RE-

UNIFY). Identifying word variants with their endings rather than word beginnings is rather more difficult and several possible solutions have been suggested. In the case where only the ending of a word is known and where the inverted file is organized by word beginnings, the conventional solution is to search the file sequentially for all terms which match the specified substring. Obviously, however, this is not a practical solution since dictionary files, even for moderate databases, can contain one hundred thousand entries or more. A much more efficient solution would be to duplicate the dictionary, writing all the words in reverse order, and processing this reversed dictionary, thus bringing back to the suffix case. An interesting discussion of such a technique may be found in Bratley and Choueka (1982). Another, more general, approach involves the system calculating a measure of string similarity between a specified query term and each of the terms in the dictionary component of the inverted file. Similar words, i.e., words with a similarity coefficient greater than some treshold values, can then be displayed at the terminal for inclusion in the query if the user so desires. Good accounts of such a procedure may be found in Adamson and Boreham (1974) and Freund and Willett (1982).

On the whole, however, conflation algorithms have inherent limitations and certain linguistic problems are common to all conflation algorithms, irrespective of their ultimate use. The general assumption (in the context of information retrieval) is that if two words have the same underlying stem then they refer to the same concept and should be indexed as such. This is not always the case, since sometimes words of the same stem, such as NEUTRON and NEUTRALIZE, need to be distinguished. Moreover, even words which are essentially equivalent may mean different things in different contexts. Thus, it is inevitable that such systems will produce errors. However, experiments have shown that the proportion of errors caused by these circumstances will not desgrade retrieval effectiveness too much (Porter,1980, Lovins 1968).

The final output from a conflation algorithm is a set of classes, one for each stem detected. A class name is then assigned to a query or a document if and only if one of its members occurs as a significant word in the query or the document. The query or document representative then becomes a list of class names (index terms or keywords). Thus, the use of conflation procedure in information retrieval systems will reduce the total number of distinct terms present, and hence a reduction in dictionary size and updating problems. In addition, if conflation is carried out on both query and document terms, similar words generally have similar meanings and thus retrieval effectiveness may be increased.

2.4 Term Weighting

2.4.1 General

In conventional retrieval systems weights are not normally assigned to designate term importance. Instead, a term is either used to identify a given item (in which case it is assumed to carry a weight 1) or it is not (in which case it is assumed to carry a weight of 0). Such a system, referred to as **a binary system**, simplifies the input process but has proved to be of limited importance; for instance, no distinction is made within the set of retrieved items, i.e. all retrieved items are considered equally important for the query. **Term weighting systems** are designed to overcome these shortcomings by allocating numerical values to each of the index terms in a query or a document reflecting their relative importance. Thus, a term with a high weight is assumed to be very relevant to the document or the query whilst a low weight indicates little relevance to the content of the document or the query. These weights can then be used to define a function which measures the similarity or closeness between each query and documents; this makes it possible to retrieve documents in the decreasing order of query-document similarity, the most similar (presumably relevant) being

retrieved first. However, the difficulty occurs in deciding how to allocate the weights and in actually calculating the values which are to be used. Various theories have been developed for the automatic assignment of term weight to the documents and queries of a collection, **including the term frequency, an inverse document frequency, the signal-noise ratio, relevance weighting and term discrimination value.** While term discrimination value will be dealt with in the following chapter, the rest are briefly discussed in the remainder of this section.

It is noteworthy that, although term weighting schemes may be used to weight either a query term or a document term or both, most of the indexing studies that have been reported involve methods for the weighting of query terms, with the documents being characterized by binary weighting (Robertson and Sparck Jones, 1976, Harper and Van Rijsbergen, 1978). Such weights are normally calculated at search time using statistics from the binary document representatives. Alternatively, terms in documents can be weighted during the indexing process. However, whether or not the latter scheme is effective in practice is still a controversial issue(Croft, 1981, Salton, 1986, Sparck Jones and Bates, 1977).

2.4.2 Term Frequency

As has been indicated, it has been argued from early on that the frequency of occurrence of a term is a useful indication of its relative importance in describing a document. One such measure proposed by Luhn (1957) assumes that the value, or weight, of a term assigned to a document is simply proportional to the term frequency, i.e. the frequency of occurrence of that particular term in that particular document. The assumption here is that the more frequently a term occurs in a document the more likely it is to be of value in describing the content of the document. Hence the weight of term k in document i (or term i), w_{ik} might be determined by,

$$w_{ik} = f_{ik}$$

where f_{ik} is the frequency of term k in document i. Salton and Yang (1973) and Sparck Jones (1973) note that in many environments this weighting system does enhance the retrieval performance as compared to binary weighting system. However, they also observe that such a weighting system sometimes does not perform as expected, especially in places where there are few high frequency terms in a collection, or when the high frequency terms are equally distributed throughout the collection. This can be easily seen from the formula, in that, it does not take into account the role of term k in any document other than document i.

2.4.3 Inverse Document Frequency

Sparck Jones (1972) introduced the concept of inverse document frequency, or IDF, weighting. Contrary to the frequency weighting introduced above, this method postulates that a good term, one which should be assigned a high weight, is one with a high occurrence frequency in a specific document, while the overall collection frequency of the term is low. The rationale behind this approach is that a high occurrence frequency in a particular document indicates that the term carries a good deal of importance in that document; a low overall collection frequency (the number of documents in the collection to which the term is assigned) indicates at the same time that the importance of the term in the remainder of the collection is relatively small so that the term can actually distinguish the documents to which it is assigned from the remainder of the collection. Thus, such a term can be considered as being of potentially greater importance for retrieval purposes. This consideration leads to a term weighting function defined as,

$$w_k = \log_2 \frac{N}{f_k} \quad or \quad w_k = \log_2 N - \log_2 f_k + 1$$

Where N is the total number of documents in the collection, f_k the number of documents in which k occurs, and w_k the weight assigned to term k. Thus as a collection frequency of a term decreases so its weight increases.

Alternative formulations of this weight in a weighted system (see section 3.1) ie,

$$w_{ik} = \frac{f_{ik}}{f_k}, \text{ or } w_{ik} = f_{ik}(\log_2 N - \log_2 f_k + 1)$$

where f_{ik} is the term frequency of term k in document i, and w_{ik} the weight of term k in document i. Tests with the IDF scheme show that it consistently produces substantial performance improvements compared to unweighted (binary) systems (Salton, 1975a, Salton and Yang 1973).

2.4.4 The Signal-Noise Ratio

The IDF discussed in the preceding subsection measures the importance of a given term by the consideration of its frequency in individual documents and collection frequency. Salton (1975a) argues that a more complete picture of term behavior may be obtained by considering the frequency characteristics of each term not only in a particular document whose terms weights are currently under consideration, but also in all other documents in the collection. One such measure which is derived from Shannon's communication theory is the so-called Signal-Noise ratio. Specifically, for a collection of n documents, the Noise of term k is defined as:

$$N_k = \sum_{i=1}^{N} \frac{f_{ik}}{f_k} \log \frac{f_k}{f_{ik}}$$

and similarly the Signal for term k as:

$$S_k = \log_2 f_k - N_k$$

The Noise N_k relates to the spread of an indexing term throughout the document collection. If a term is evenly distributed throughout the collection (i.e. if it is a non-specific term), then the Noise is maximized, whereas if it appears in only one document with its total frequency then the Noise is zero. A weighting function based on the Signal-Noise parameter may be

defined as:

$$w_k = \frac{S_k}{N_k}, \text{ or } w_k = f_{ik} S_k$$

It is suggested that even the simple ranking in order of decreasing Signal is sufficient to indicate the 'better' index terms. An investigation between Signal-Noise ratio and term frequency reveals that a high Signal value often implies high document frequency and that high Noise relates to high collection frequency terms; an analogous relation holds for low Signal, Noise and frequency values (Salton and McGill, 1983)

2.4.5 Probabilistic Approach

All the weighting schemes discussed above are based on the frequency of occurrence of terms both within individual documents and in the document collection as a whole. Another set of weighting scheme relates to recent work on probabilistic models (Van Rijsbergen, 1979) that use not only the term frequency characteristic but also the relevance properties of terms. A number of workers have proposed interesting measures based on this idea. Salton and Wu (1981) propose a theory of precision weighting, by which each query term is assigned a weight in accordance with its relative frequency of occurrence in relevant and non-relevant documents. The expression they give for term precision (term importance factor) is :

$$L_k = \frac{r_k/(R - r_k)}{(n_k + r_k)/(N - n_k - (R - r_k))}$$

Here, L_k is the term precision for term k, n_k is the number of documents containing term k out of a total of N documents, r_k is the number of relevant documents containing term k out of the total of R relevant documents; hence $n_k - r_k$ is the number of non-relevant documents containing k out of a total of N-R non-relevant documents. An appropriate term-weighting function for term k in document i can then be defined as :

$$w_{ik} = f_{ik} L_k$$

It has to be noted that the application of this formula requires the initial estimation of the relative occurrence of the search term in relevant and non-relevant documents. Yu et al (1982) suggest that this may be obtained using relevance feed back methods with inverse document frequency weights as the initial set. The term precision factor, L_k , indicates the difference between actually retrieved results and how effective a search could have been; the search efficiency is improved by using the new set of weights as a basis for a second search and so on, which should retrieve a greater proportion of the relevant documents. This suggests its use on a second or subsequent search, when some relevant documents have been identified by the first iteration.

Robertson and Sparck Jones (1976) have also developed a theory along the same lines, i.e., the use of relevance information as a basis for the weighting of query terms. The weighting function addressed here differs from that of the previous one in that this is directly used in weighted retrieval; the precision weighting is not used directly in weighted retrieval, instead the documents retrieved at a given level of coordination are further ordered by means of the precision weight. Using probability theory and the assumption that the occurrences of index terms in documents were statistically independent of each other, they were able to provide a theoretical rationale for the use of a term weight,

$$w_i = \log \frac{p_i(1 - q_i)}{q_i(1 - p_i)}$$

Where w_i is the weight assigned to term i, p_i and q_i refer to the probability of the i-th term occurring in a relevant and in a non-relevant document respectively. These probabilities are difficult to calculate directly. However, they can be approximated in a number of different ways. If there is a complete information about the relevant and non-relevant documents in the collection, then one can estimate p_i by r/R and q_i by (n-r)(N-R) (Van Rijsbergen, 1979), where once again r and R are as in the above convention. Thus the weights can be expressed as:

$$w_i = \log \frac{r(R - r)}{(n - r)/(N - n - r - R)}$$

It is interesting to note that, although the starting points are different, the anti-log or $exp(w_i)$ in this last expression is simply the formulae for calculating the term precision factor L_i , derived by Salton and his co-workers.

A modified form of this last expression which would be used to produce an overall weight for each query is :

$$\sum d_i \log w_i$$

where d=1 if the term is present, and d=0 otherwise, and where the summation is over all of the terms in the query (Van Rijsbergen, 1979).

A number of studies (Sparck Jones and Bates, 1977, Yu et al, 1982) have indicated these weights produce excellent retrieval performance, specially in the so-called retrospective experiments when full relevance information is available (i.e., where r and R are known for each term and query).

In the case where complete relevance data is not available, the use of partial relevance information obtained by retrieving documents on the basis of the query and presenting them to the user for judgement as relevant or non-relevant, is suggested (Sparck Jones, 1979a, Robertson and Sparck Jones, 1976, Yu et al, 1982).

However, such relevance feedback techniques cannot be used until the user has had the chance to provide some relevance information to the system. This question has been addressed by Croft and Harper (1979) who propose a probabilistic model for the initial search, based on the term-independence assumption (**that index terms occur independently in the relevant and non-relevant documents**). They argue that in the situation where no information about relevant documents is available, it is possible to assume that all the query terms have equal probabilities of occurring in the relevant documents. **The occurrence of a term in the non-relevant documents, on the other hand, can be estimated by its occurrence in the entire collection.** These two assumptions cor-

respond to setting $\frac{p_i}{1-p_i}$ equal to a constant, and estimating q_i by n_i /N where n_i is the number of documents in which the term i occurs and N is the size of the collection. Therefore, the expression above may be reduced to :

$$\sum d_i \log \frac{c(N - n_i)}{n_i} = \sum d_i \log C + \sum d_i \log \frac{N - n_i}{n_i}$$

where $c = \frac{p_i}{1-p_i}$ the constant resulting from the first assumption. It is clear the above that the first part of this expression corresponds to the number of terms in common between the document and the query, when the binary representation is considered. While the second part, for large N (i.e. if the number of documents in the collection is large), is similar to the IDF weight.

Thus this probabilistic model indicates that in the absence of relevance information, the best function for ranking the documents is a combination of a simple match and a match using inverse document frequency weights. Croft and Harper report that, for the collection tested, the method has produced results that were significantly little better than simple matching, inverse document frequency or the cosine similarity function, especially when the value of p_i is set to 0.9 (close to 1.0) in which case the constant c becomes very large (i.e. c=9).

Although the relevance weighting is very successful, in practice due to the the large number of parameters that must be estimated, it is not possible to relate it as a method to any of the other weighting methods in use (Robertson, 1977a, Salton and Wu, 1981). However, an interesting possibility raised by its use is that of changing the formulation in order to produce an optimal query. This is done by considering the query terms in relation to the relevance of documents retrieved in the first iteration of the process. Those terms which appear in many of the relevant documents, but do not appear in the query, may be added in to the query statement, and those terms that have caused non-relevant elements to be retrieved may be assigned lower value weights, for the second iteration. A good account of such

techniques may be found in Robertson (1986).

Chapter 3

The Discrimination Model

3.1 Theory

An elegant mathematical model for the indexing process which provides facilities for selecting and weighting index terms is the term discrimination model introduced in section 2.2. Much has been written on this theory since the early 70s when the idea was first reported (see the reference list for a selection of these papers). As mentioned earlier, this model is designed to rate a given index term in accordance with its usefulness as a discriminator among the documents of a collection (i.e. how well they are able to discriminate the documents of a collection from each other). In addition, it offers a reasonably accurate physical interpretation of the indexing process (Salton, 1975a).

The term discrimination model takes as its basis the idea of document space configuration, where a document may be considered as a vector the elements of which are the weights of every indexing term to describe the document, and where retrieval is considered as a vector matching operation between a query vector (defined, similarly, as a vector the elements of which

are the weights of the terms used to describe the query) and a document vector. Consider a collection D of n documents indexed by a set of m terms. A particular document D can be represented by a vector:

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

where d_{ij} represents the weight, or degree of importance of the j-th term. The d_{ij} may be weighted according to their importance, for instance, the term frequency of term j in document i (Salton, 1975b) or the weight proposed by Harter (1975), or unweighted with weights restricted to the values 0 or 1, as in the binary system. These vectors can then be represented as belonging to a m-dimensional vector space.

The relative distance between the vectors may be preserved by normalizing all vector lengths to one, and considering the projection of the vectors onto the envelope of the space represented by the sphere. In that case, each document may be represented by a single point whose position is specified by the location where the corresponding document vector touches the envelop of the space. It is then possible to compute a similarity coefficient, $S(D_i, D_k)$ between any two documents by comparing the corresponding vector pairs, which reflects the degree of similarity or closeness between the two documents.

A typical similarity measure between documents D_i and D_j might be the **dot product**, which is given by the product of the two vectors, i.e.

$$S(D_i, D_k) = \sum_{j=1}^{M} d_{ij} d_{kj}$$

In the case of unweighted terms (binary vectors), this corresponds to just the number of terms in common between the two vectors, whereas for weighted vectors it is the sum of the products of corresponding term weights.Another well-known vector similarity measure is the **cosine coefficient** , in which the dot product is normalized by the square root of the sums of the squares

of the corresponding vector elements, i.e.

$$S(D_i,\ D_k) \ = \ \frac{\sum_{j=1}^{M} d_{ij} d_{kj}}{\sqrt{\sum_{j=1}^{M} d_{ij}^2 \sum_{j=1}^{M} d_{kj}^2}}$$

This coefficient has been extensively used, and seems to give better results in practice. Other similarity functions include the dice and overlap coefficients, and the euclidean distance.

The existence of the term sets representing the various documents, and the computation of similarity measures between documents serves to define a document space for a given collection. In such a space two documents appear in close proximity when their similarity coefficient is large (i.e. when their corresponding term vectors are quite similar); contrariwise, documents exhibiting little similarity are widely separated in the document space. Furthermore, as there is a one-to-one relationship between the set of index terms used to characterize a collection of documents and the space generated by the indexed documents, the problem of choosing an effective term set may be transformed into one of producing a useful document space (Salton, et al, 1975a). In this context, a clustered document space where all documents exhibit somewhat similar term sets, is not a desirable situation for retrieval since it is difficult to distinguish documents from one another. On the other hand, when the indexing leads to an even distribution of documents in the space, the separation of relevant from non-relevant documents is more difficult to achieve. The ideal space, therefore, is one in which all similar documents (presumably jointly relevant to a certain user query) are clustered tightly together, thus insuring that they could be retrievable jointly in response to the corresponding queries, while at the same time being quite distinct from dissimilar documents. However, such a space is rarely encountered in practice, i.e., in most cases the documents do not produce useful clusters in the document space. Moreover, such a space based on relevance clusters is difficult to construct since the set of documents jointly relevant to the queries is unknown at indexing time.

In such circumstances, Salton and his co-workers claim that the next best

space configuration may then be a separated document space in which each document exhibits the widest possible separation from all its neighbours. In particular, they argue that maximum separation may be achieved through minimizing the total similarity of the space, i.e. minimizing

$$f = S(D_i, D_k), \quad i \neq j$$

The rationale behind this apparently counter-intuitive method is that when adequate separation is achieved between the documents in the document space, it is possible in principle to retrieve a given item in response to a query without at the same time retrieving its immediate neighbours, thereby insuring high precision search outputs. In cases where several different relevant items for a given query are located in the same general area of the space, it may then also be possible to retrieve many of the relevant items while rejecting most of the non-relevant, hence receiving high recall and precision.

To this end, the term discrimination model is suggested to satisfy the requirements of the above discussed indexing theory. Specifically, the value of an index term is assumed to depend on its ability to effect separation in the space when assigned to documents of the collection. A useful term, a good discriminator, is one which renders the documents less similar to each other when assigned as an index term to the documents of a collection (i.e. its assignment decreases the space density or spreads out the space when assigned to the documents). A poor discriminator, on the other hand, increases the space density (compresses the space) when assigned as a content identifier, whereas an indifferent discriminator leaves the space unchanged. Hence, it has been suggested that the inclusion of a good discriminator into the document vectors will facilitate the retrieval and query construction, and that such an inclusion can be effected by including weights based on the discriminant values of the indexing terms in the document vectors.

For each potential index term, a discrimination value (DV) can be computed as the difference in space density before and after assignment of that

term. The greater the difference in space densities, the more the space will be spread after assignment of a given term, and therefore the better that term will function as a discriminator. The density or compactness, Q, of the document space is computed as the average pair-wise similarity between all pairs of distinct documents. i.e.

$$Q = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i \neq j}}^{N} S(D_i, D_k)$$

Where N refers to the total number of documents in the collection. In order to determine whether a particular index term has had a good or bad effect on the document space it is necessary to consider the average similarity for the space when that term is not used as an indexing term. Thus it is necessary to calculate:

$$Q_k = \frac{1}{N(N-1)} \sum_{\substack{i=1 \\ i \neq k}}^{N} \sum_{\substack{j=1 \\ j \neq i \\ j \neq k}}^{N} S(D_i, D_j)$$

If term k represents a good discriminator then it spreads out the document space while its removal compresses the document space and makes the documents more similar each other: in this case $Q_k > Q$. Contrariwise, if term k is a poor discriminator its removal is likely to decrease the average pairwise similarity in which case $Q_k < Q$. Accordingly, q_k is approximately equal to Q for the indifferent terms, i.e. those where the removal of the term has little a marginal or no effect on the overall collection similarity.

A discrimination value can now be computed for each term k, as a function of the value $(Q_k - Q)$ which assigns positive weights to the good discriminators, a negative ones to the poor discriminators, and a zero value to the indifferent ones. In practice, this function is frequently normalized to provide a convenient base for comparison; i.e.

$$DV_k = \frac{Q_k - Q}{Q}$$

As indicated in the previous chapter, such term discrimination value measurements can be used as an indexing aid by choosing as document identifiers

terms which exhibit sufficiently high discrimination values. Furthermore, these values can be incorporated into a term weighting function by defining,

$$w_{ik} = f_{ik} DV_k$$

Where w_{ik} is the weight that is assigned to term k, f_{ik} the frequency of occurrence of term k in document i. Such a weighting system has shown to produce excellent results in practice (Salton and McGill, 1983).

3.2 Discrimination Value and Document Frequency

In order to translate the discrimination value model into a possible theory of indexing it is necessary to examine the frequency distribution of good and bad discriminators. Several experiments have been carried out to study the relationship between discrimination values and frequency characteristics of the terms in document collections (Salton et al, 1975b, Salton, 1975a, Willett, 1985).

The method used to study this relationship was originally suggested by Salton and has been used by almost every one working in this field: it is as follows. Firstly, the discrimination values are calculated for all of the indexing terms (the various ways which have been suggested for the calculation of discrimination values are described in the next section). Having found the discrimination values, the terms are then sorted into order of decreasing discrimination value, so that the most highly discriminating term is given rank 1 and so on. The average ranks are then calculated for all of the terms appearing at a particular frequency. Finally a graph of average rank against term frequency is produced to show the relationship between the frequency of occurrence and the average discrimination ranks.

Experimental results, in particular when the cosine similarity measure is used, reveal that: the best discriminators, those with positive discrimination values, exhibit medium document frequencies, that is terms that are assigned

neither to too many nor to too few documents. Broad terms, terms with high frequency, show negative discrimination values. Narrow terms, terms that appear rarely in the collection, result in discrimination values close to zero. Specifically, Salton et al (1973) observe that the discrimination value for terms having low document frequencies are in excess of m/2 where there are m indexing terms. High frequency terms, exceeding n/10, where n is the number of documents, have average discrimination value of about m. The best discriminators, with document frequencies between n/100, and n/10, have average discrimination value ranks below m/5.

However, Willett (1985) noted that the identification of the intermediate frequency terms as the most discriminating is crucially dependent upon the type of inter-document similarity measure that is used for the calculation of the discrimination values. He provided both mathematical analysis and the results obtained from the experiments carried out. The range of similarity functions examined include the Cosine, Dice, Jacard or Overlap coefficients, the dot product and the euclidean distance. When the relationship between discrimination values and collection frequencies is studied, as suggested above, it was found that the use of similarity functions such as the Dice, Jacard coefficients produced similar results to that of the cosine function, whereas quite different results were obtained from the dot product and euclidean distance coefficients. More specifically, the results obtained from the use of dot product suggested that the most discriminating terms were those of infrequent occurrences in the collection, whereas the results obtained from the use of euclidean distance identified frequent terms as the most discriminating ones. The identification of medium frequency terms as the most useful for indexing is also in disagreement with the IDF weighting system, as described below.

3.3 Discrimination Value and IDF Weighting

By and large the most discussed and used weighting systems are the IDF and term discrimination value systems (Robertson and Sparck Jones, 1976, Salton and McGill, 1983). While both systems produce comparable and excellent results in practice, there is still an unresolved conflict between the two methods. The major difference between them is that while the IDF method assumes that the best indexing terms are those that occur with smallest frequency, the term discrimination method identifies medium frequency terms as best for indexing. A number of workers have attempted to provide explanations for such conflict.

Salton and Yang (1973) observe that there are a number of frequency groups into which the indexing terms may be divided which exhibit useful characteristics for retrieval performance. They note that good indexing terms are those of medium frequency terms with a somewhat skewed frequency distribution, i.e., high frequency in some documents and low in others. They also point out that medium frequency terms with flat distribution, i.e., terms occurring with equal frequency in all documents, cannot be used to retrieve documents precisely: they only differentiate the documents in which they occur from the rest. In addition, terms occurring in only few documents with a very skewed distribution (because they are more likely to produce high matching with a query), and those terms with very low overall frequencies (because of their rarity, a match will isolate a few documents from the bulk of the remainder) are suggested to be good indexing terms. Thus, in simple terms, they suggest that not all terms with medium or low frequency frequency are good discriminators.

Yu and Salton (1977) take a rather different approach and argue that for each individual query a given indexing term will have a wide range of possible relevance values; for each specific query the best indexing terms are those having low frequency, whereas when the relevance is averaged over

a set of queries it seems that the terms with medium frequency are best. Thus the difference lies in whether one is considering overall effectiveness or effectiveness in a particular case.

Salton and Wu (1981) explain this difference from the utility theory model viewpoint. They claim that the utility theory model leads to an optimal term weight behaviour which is in agreement with the discrimination value model: they base this conclusion on the similarity between the graphs of frequency against document weight produced by the two theories. They also argue that, on the basis of the analytic results for the utility model, both IDF and discrimination value systems produce similar term weights when the point of maximum weight occurs at f=1, for such a point, the argument goes on, w will decrease monotonically with f. This observation is in accord with the IDF assumption that the greatest weight are given to the least frequent terms (f=1) and the least weight to the term with the greatest frequency.

One can also argue, based on the discussion in the preceding subsection, that such difference may result from the way in which the inter-document similarity is calculated. After all, the use of similarity functions such as the dot product produces results which are similar to IDF systems.

However, there is no clear explanation, from the experiments reported to date, on this disagreement. In fact, generally speaking, this has provided a basis for the controversial issue as to which weighting system should be used in order to gain maximum benefit. Salton and his co-workers (Salton and Yang, 1973, Salton and McGill, 1983) have long advocated the use of both term discrimination value and inverse document frequency depending on the level of recall and precision required. Sparck Jones and Bates (1977) argue that the IDF weighting is preferable since it appears to be able to do all that the term discrimination value weighting does, i.e., the performance derived from IDF is very similar to that from using discrimination value, and its computational requirements are much less than those of the term

discrimination model.

3.4 Discrimination Value Computation

The practical implementation of the discrimination value method implies the calculation of the measure of similarity between each pair of documents in the collection before and after the deletion of a particular term. The various algorithms proposed to calculate these values are briefly described in the remainder of this section.

3.4.1 Computation Using the Cosine Coefficient

As indicated in previous sections, the most extensively used measure of similarity between some pairs of documents D_j and D_k is the cosine function defined by,

$$\cos(D_j,\ D_k) = \frac{\sum_{i=1}^{M} d_{ji} d_{ki}}{\sqrt{\sum_{i=1}^{M} d_{ji}^2 \sum_{i=1}^{M} d_{ki}^2}}$$

The space density, which is measured in terms of Q (compactness) in section 3.1, can then be formulated as:

$$Q = \frac{1}{N(N-1)} \sum_{j=1}^{N} \sum_{k=1}^{N} \cos(D_j, D_k)$$

and accordingly,

$$Q_i = \frac{1}{N(N-1)} \sum_{j=1}^{N} \sum_{k=1}^{N} \cos(D_j^i,\ D_k^i)$$

The measure of the discriminating power of an individual term i (DV_i) can then be obtained by

$$DV_i = \frac{Q_k - Q}{Q}$$

Figure 1 shows an algorithm for such a procedure in a PASCAL-like notation. In practice, however, it is very difficult to calculate Q, Q_i and then DV_i values using this algorithm, since the number of computations involved

is very large. For a collection of N documents and M index terms such algorithm involves the computation of N(N-1)/2 coefficients for each of the M index terms, and N(N-1)/2 initial set of coefficients for the evaluation of Q. Thus the complexity of the algorithm is of order $O(MN^2)$ In free text retrieval system, where the value of M may even exceed that of N, the computation of term discrimination value using this algorithm is not feasible.

$Q := 0;$
FOR $i := 1$ *TO* $N - 1$ *DO*
  FOR $j := i + 1$ *TO* N *DO*
    $Q := Q + \cos(d_j, d_k);$
FOR $i := 1$ *TO* M *DO*
  BEGIN
    $Q_i := 0;$
    FOR $j := 1$ *TO* $N - 1$ *DO*
      FOR $k := j + 1$ *TO* N *DO*
        $Q_i := Q_i + \cos(d_j^i, d_k^i);$
    $DV_i := \frac{(Q_i - Q)}{Q}$
  END.

Figure 1. Algorithm A.

This problem has been addressed by a number of workers who suggested various ways of reducing the problem to a computationally feasible level. For the purpose of our discussion, such attempts are reviewed as early and recent approaches, where the former refers to relatively early work on reducing the computation by approximating the actual values (i.e. using centroid vectors), and the latter refers to quite recent research results on the feasibility of the original idea of exact values (i.e. using the actual document vectors).

In order to bypass the difficulty of the practical implementation of algorithm A, previous work on term discrimination value takes quite a different approach for the calculation of the values. This involves the use of an artificial representative point in the space, a centroid (Salton and Yang, 1973, Salton et al, 1975). For a collection of N documents, each element of the centroid C may then be defined as the arithmetic average of the same elements in the corresponding document vectors, that is

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

Having computed the centroid, the computation of the similarity between all pairs of documents may be replaced with the computation of the similarity between every document and the centroid. This obviously reduces the number of operations needed to calculate the space density Q , from N^2 in the previous case to N, which means a lot especially for big collections (large values of N).

In particular, using the centroid method, the cosine similarity function can be conveniently expressed as

$$\cos(C,\ D_j) = \frac{\sum_{i=1}^{M} c_i d_{ji}}{\sqrt{\sum_{i=1}^{m} d_{ji}^2 \sum_{i=1}^{M} c_i^2}}$$

and similarly,

$$Q = \frac{1}{N} \sum_{j=1}^{N} \cos(C,\ D_j)$$

The value of Q is strictly greater than zero, since the individual values of $\cos(C,\ D_j)$ will not be zero for all j due to the definition of the centroid. Similarly, the value of Q will not exceed one since Q is simply the average value of $\cos(C,\ D_j)$.

Accordingly, the Q_i values are obtained from

$$Q_i = \frac{1}{N} \sum_{j=1}^{N} \cos(C^i,\ D_j^i)$$

Where C^i and D^i_j refer to the centroid and the j-th document with the i-th term deleted. The discrimination value DV for term i may then be obtained simply from $(Q_i - Q)/Q$. This algorithm may then be described as shown in Figure 2. It can be seen from the algorithm that only N(M + 1) document centroid similarities in all need to be calculated.


```
FOR i := 1 TO M DO
  BEGIN
    c_i := 0;
    FOR j := 1 TO N DO
      c_i = c_i + d_ji;
    c_i := c_i / N
  END
Q := 0;
FOR i := 1 TO N DO
  Q := Q + cos(C, d_j);
FOR i := 1 TO M DO
  BEGIN
    Q_i := 0;
    FOR j := 1 TO N DO
      Q_i := Q_i + cos(C^i, d^i_j);
    DV_i := (Q_i - Q) / Q
  END.
```

Figure 2. Algorithm B

Crawford (1975) analyses this algorithm and particularly notes that due to the massive computation involved, such an algorithm is not practically feasible. He then gives a number of useful refinements that can be made to the algorithm which reduces the number of individual computations by a significant amount. The first of these involves the reduction in the number of iterations required to calculate the similarities. Crawford notes that during

the calculation of each Q_i value in algorithm B, a summation is made over all of the $\cos(C^i, D^i_j)$ values even though most of the documents will not contain term i. However, the $\cos(C^i, D^i_j)$ values for those documents not containing i, will be virtually the same as the $\cos(C, D_j)$ values that were used to compute Q initially. Thus, in order to avoid these unnecessary iterations, he suggests an alternative way of calculating Q_i , which is by modifying Q only for those documents in which term i appears, i.e.

$$Q_i = Q - \cos(C, D_j) + \cos(C^i, D^i_j)$$

On the practical implementation of this refinement (i.e. the necessary step to be able to determine exactly in which documents each term occurs), he suggests one of the following two ways. The first possibility is to maintain the $N \times M$ document term matrix, where locating the non-zero values in column i would give the documents (rows) in which term i occurs. However, this is suggested to be both impractical and prohibitively expensive in terms of computer storage, as most documents have large values of M and N. The second possibility is the use of an inverted file. An inverted file contains a set of lists, one for each of the terms in the indexing vocabulary that is used to characterize the documents, such that the i-th list contains the identifiers of those documents containing the i-th term.

The other possible improvement to the algorithm, suggested by Crawford, is in computing the cosine values. He notes that the value of cosine will be computed several times for each term i in a document j, in algorithm B. This calculation requires 3M computations of which only 3 will have changed values. Hence all the rest of the 3(M-1) computations are redundant. This redundancy can be avoided if various partial values are stored for the purpose of the calculation. In particular, by computing and storing the values of,

$$DOTPRODC_j = \sum_{i=1}^{M} c_i d_{ji},$$

$$SUMSQC = \sum_{i=1}^{M} c_i^2,$$

$$SUMSQD_j = \sum_{i=1}^{M} d_{ji}^2$$

In which case, $\cos(C,\ D_j), \cos(C^i,\ D_j^i)$ can be expressed as,

$$\cos(C, D_j) = \frac{DOTPRODC_j}{\sqrt{SUMSQC * SUMSQD_j}},$$

$$\cos(C^i, D_j^i) = \frac{DOTPRODC_j - c_i d_{ji}}{\sqrt{(SUMSQC - c_i^2)\,(SUMSQD_j - d_{ji}^2)}}$$

Figure 3 below shows the algorithm which results from implementation of the refinements in the foregoing discussion. As can be seen from the algorithm, although some additional computations are introduced in calculating the constants and producing the inverted file, this algorithm saves a significant amount of computer storage and is also faster in operation. It has, however, been noted that the DV_i values computed using this algorithm are not exactly identical to that obtained from algorithm B (Willett, 1985).

Recent approach

Willett (1985) has recently described a modified version of algorithm A which involves the calculation of only $O(N^2n)$ similarity coefficients, where n is the mean number of indexing terms assigned to each of the documents in a file. The algorithm takes as its basis two observations in algorithm A. Firstly, contributions are made to all of the MQ values as each inter-document similarity coefficient is calculated, rather than by calculating Q_i , and hence DV_i for each term in sequence. Secondly, improvements are achieved if a different approach is taken to the evaluation of $Q_i - Q$. Instead of calculating Q as the sum of the $cos(D_j,\ D_k)$ values and Q_i as the sum of $\cos(D_j^i,\ D_k^i)$ values independently, and then the difference for DV_i (which is the case in algorithm A), the difference of the sum may be replaced with the sum of the differences, i.e.

$$\begin{aligned} Q_i - Q &= \sum \cos(D_j^i, D_k^i) - \sum \cos(D_j,\ D_k) \\ &= \sum \left(\cos(D_j^i, D_k^i) - \cos(D_j,\ D_k)\right) \end{aligned}$$

Therefore, if

$$DOTPRODJK = \sum_{i=1}^{M} d_{ji}d_{ki},$$

$$SUMSQD_j = \sum_{i=1}^{M} d_{ji}^2,$$

$$SUMSQD_k = \sum_{i=1}^{M} d_{ki}^2 \qquad (3.1)$$

for all pairs of documents D_j and D_k , then, according to the foregoing,

$$\cos(D_j, D_k) = \frac{DOTPRODJK}{\sqrt{SUMSQD_j SUMSQD_k}}$$

and the difference $\cos(D_j^i, D_k^i) - \cos(D_j, D_k)$ may be expressed as:

$\frac{DOTPRODJK - d_{ji}d_{ki}}{\sqrt{(SUMSQD_j - d_{ji}^2)(SUMSQD_k - d_{ki}^2)}} - \cos(D_j, D_k) \ldots (DIFF1)$ (when the term i appears in both D_j and D_k;

$\frac{DOTPRODJK}{\sqrt{(SUMSQD_j - d_{ji}^2)SUMSQD_k}} \ldots (DIFF2)$ (when the term i appears in D_j only); $\frac{DOTPRODJK}{\sqrt{SUMSQD_j(SUMSQD_k - d_{ki})}} \ldots (DIFF3)$ (when the term appears in D_k only);

and 0 . . . (when the term i does not appear in both).

Figure 4 depicts the algorithm for such a procedure. As can be seen from the algorithm this procedure involves the calculation of only N(N-1)/2 $\cos(D_j, D_k)$ values, and M increments for each $\cos(D_j, D_k)$ values, and hence it seems to be of order $O(MN^2)$ complexity. However, it should be noted that the number of increments occasioned by each similarity coefficient will be equal to the number of distinct terms assigned to the two documents since in the case when the i-th term does not appear in either documents (which is in fact by far the most frequent case), $\cos(D_j^i, D_k^i)$ and $\cos(D_j, D_k)$ are identical and cancel each other out and hence may be neglected. In that case, the total number of increments will be proportional to tN , where t is the mean number of indexing terms assigned to each document, thereby

making this algorithm computationally feasible for the small-sized document test collections currently used in information retrieval research.

More recent work by El-Hamdouchi and Willett (1988a) proposes an improved algorithm for the calculation of exact term discrimination values. The complexity of the algorithm is of order $O(Nn^2)$, for a collection of N documents, each of which has been assigned an average of n terms. This work takes as its basis the work of Voorhees (1986) which involves the use of the group average hierarchic agglomerative clustering method for automatic document classification. In particular, for any two clusters, say A and B, with N1 and N2 documents respectively, the calculation of the N1xN2 average inter-document similarity coefficients for all pairs of documents, one from each cluster, can be replaced by the calculation of a single inter-centroid similarity. Where the elements of a centroid for each cluster are defined as a linear combination of the elements in the corresponding document vectors in that centroid. That is,

$$a_i = \sum_{j=1}^{N1} w_j D_j$$

Where a_i is the i-th element of A, w_j is the weight of the j-th document, and D_j is the j-th document in A. Similarly, for B,

$$b_i = \sum_{j=1}^{N2} w_j D_J$$

where D_j is the j-th document in B.

The dot product of the two centroids (DOTPRODAB, for short) may be written as,

$$\begin{aligned} DOTPRODAB &= \sum_{j=1}^{N1} w_j D_j \sum_{k=1}^{N2} w_k D_k \\ &= \sum_{j=1}^{N1} \sum_{k=1}^{N2} w_j w_k DOTPRODJK \end{aligned}$$

If a weighting such as w = 1/SUMSQJ , where SUMSQJ= $\sum_{i=1}^{M} d_{ji}$ is used, the above expression becomes,

$$DOTPRODAB = \sum_{j=1}^{N1} \sum_{k=1}^{N2} \frac{DOTPRODJK}{\sqrt{SUMSQD_j SUMSQD_k}}$$

$$= \sum_{j=1}^{N1} \sum_{k=1}^{N2} \cos(D_j,\ D_k)$$

El-Hamdouchi and Willett then start with the assumption that A=B=C, where C is the centroid of the entire document collection, i.e.

$$c_i = \sum_{j=1}^{N} w_j Dj$$

where c is the i-th component of C. In that case the above expression can be written as

$$DOTPRODCC = \sum_{j=1}^{N} \sum_{k=1}^{N} w_j w_k DOTPRODJK$$

Which can be formulated as,

$$DOTPRODCC = 2 \sum_{\substack{j,k=1 \\ j<k}} w_j w_k DOTPRODJK + \sum_{j=1}^{N} w_j^2 SUMSQD_j$$

Note that the first expression to the right of this last equation is nothing but the sum of all of the inter-document similarities (the compactness Q). Rearranging this equation,then, we get,

$$2Q = SUMSQC - \sum_{j=1}^{N} w_j^2 SUMSQD_j$$

where SUMSQC=DOTPRODCC, and similarly it is possible to obtain Q_i as

$$Q_i = SUMSQCI - \sum_{j=1}^{N} w_j^{i^2} SUMSQD_j^i$$

where SUMSQCI and $SUMSQD_j^i$ refer to the sum of the squares for the centroid and for the j-th document respectively when the i-th term has been deleted from the indexing vocabulary, and where w_j^i is the corresponding weight when this deletion has occurred. Therefore, after the obvious substitution, the expression for calculating the DV_i values can be written as,

$$\begin{aligned} DV_i &= SUMSQCI - SUMSQC - \sum_{j=1}^{N} w_j^{i^2} SUMSQD_j^i + \sum_{j=1}^{N} w_j^2 SUMSQD_j \\ &= SUMSQCI - SUMSQC - N + N \\ &= SUMSQCI - SUMSQC \end{aligned} \qquad (3.2)$$

It is clear from the foregoing derivation that, when the weighting described above is used, the N(N-1)/2 inter-document similarities which are involved in the calculation of DV values may be replaced by a single computation of the inter-centroid similarity.

Substituting SUMSQC and SUMSQCI with the actual values, we get,

$$\begin{aligned} DV_i &= \sum_{\substack{k=1 \\ k \neq i}} \left(c_k^{i^2} - c_k^2 \right) - c_i^2 \\ &= \sum_{\substack{k=1 \\ k \neq i}} ((c_{ik} + c_k)(c_{ik} - c_k)) - c_i^2 \end{aligned} \tag{3.3}$$

For each term $k \neq i$, c_k, can be expressed as,

$$\begin{aligned} c_k &= \sum_{j=1}^{N} w_j d_{jk} \\ &= \sum_{D_j \in KI} w_j d_{jk} + \sum_{D_j \in KNI} w_j d_{jk} \end{aligned}$$

where KI and KNI refer to the set of documents that are indexed by both k and i and that are indexed by k but nor i respectively. Similarly,

$$c_{ik} = \sum_{D_j \in KI} w_j^i d_{jk} + \sum_{D_j \in KNI} w_j^i d_{jk}$$

However, in KNI $w_j^i d_{jk} = w_j d_k$, since these documents are not indexed by i. Hence the difference $(c_k^i - c_k)$ is always zero for all documents indexed by i. In which case, the expression for DV_i can be formulated as,

$$DV_i = \sum_{\substack{k=1 \\ k \neq i}} (DIF_k(DIF_k + 2c_k)) - c_i^2$$

where for each term k, $k \neq i$,

$$\begin{aligned} DIF_k &= \sum_{D_j \in KI} w_j^i d_{jk} - \sum_{D_j \in KI} w_j d_{jk} \\ &= \sum_{D_j \in KI} (w_j^i - w_j) d_{jk} \end{aligned} \tag{3.4}$$

In the case where the indexing system is binary ($d_{ij} = 1$ for every D_j e KI, since every document in KI is indexed by i), w_j can be written as, $w_j = \frac{1}{SZ_j}$,

where $SZ_j = SUMSQJ$, the number of index terms describing document j. When the i-th term is deleted, the number of terms used to describe D_j will decrease by 1, hence $w_j = \frac{1}{(SZ_j - 1)}$. Therefore,

$$DIF_k = \sum_{D_j \in KI} \left(\frac{1}{\sqrt{SZ_j - 1}} - \frac{1}{\sqrt{SZ_j}} \right)$$

This procedure may then be described as shown in algorithm E, Figure 5. Note that, the calculation of DV_i values using this algorithm involves only those terms which co-occur with i, rather than considering all the M terms. Such terms may be identified efficiently by using the inverted file discussed above.

3.4.2 Computation Using the Cover Coefficient

Another technique to calculate the discrimination values, suggested quite recently by Can and Ozkarahan (1987), involves the use of a new concept, called the Cover coefficient, instead of the cosine coefficient to measure the separation of the documents in the document space. For a collection of N documents, indexed by M terms, the Cover coefficient for the collection may be visualized as an $N \times N$ matrix C, where each element, c_{ij}, of C indicates the extent to which document i is covered by document j. The elements of C, c_{ij} , are defined as

$$c_{ij} = \sum_{k=1}^{N} \left(S_{ik} \times S'^{T}_{kj} \right) \quad 1 \leq i, j \leq N$$

where S_{ij} (significance of term j in document i) and S'_{ij} (significance of document i in term j) are defined as,

$$S_{ij} = \frac{d_{ij}}{\sum_{k=1}^{M} d_{ik}} , S'_{ij} = \frac{d_{ij}}{\sum_{k=1}^{N} d_{ki}}, \quad 1 \leq i \leq N, \; 1 \leq j \leq M$$

and S'^{T} implies the transpose operation over the matrix S'. Thus, if

$$SUMROW_i = \sum_{j=1} Md_{ij}, \; and \; SUMCOL_j = \sum_{k=1}^{N} d_{kj}$$

then,

$$c_{ij} = \frac{1}{SUMROW_i} \sum_{k=1}^{M} \frac{d_{ik} d_{jk}}{SUMCOL_j} , 1 \leq i, j \leq N$$

As indicated above, each element c_{ij} of the matrix C indicates the extent of coverage of document i by document j. Hence the extent to which a certain document i is covered by itself is denoted by the diagonal element of C, i.e. c_{ii} , and it is known as the **uniqueness or decoupling coefficient** of document i ($DCPL_i$, for our purpose). For every document i, the values of $DCPL_i$ fall in the range $0 < DCPL_i < 1$. It is noteworthy, however, that the diagonal elements of C, c_{ii} , are not related to $S(D_i, D_i)$, since $S(D_i, D_i)$ is always equal to one. $DCPL_i > 0$ since according to the definition of C, a document is always covered by itself. The value of $DCPL_i$ will be one if none of the terms of document i is used by any other document (in which case the inner sum above will simply cancel each other and the outer sum is the reciprocal of the row some). In such cases, document i is said to be completely unique or "decoupled" from the other documents of the collection. This concept of decoupling coefficient was originally introduced for document clustering purposes. In particular, it was noted that the sum

$$DCPL = \sum_{i=1}^{N} DCPL_i = \sum_{i=1}^{N} c_{ii}$$

is nothing but the number of clusters (NC) within a collection (Can and Ozkarahan, 1984). Can and Ozkarahan (1987) suggest the use of the average decoupling coefficient of the collection (DCPL/N) for calculating term discrimination values in the following manner.

In term discrimination theory, the discriminating power of a term is evaluated by its ability to spread out the document space when used as an index term. In previous work, this is quantified by the difference in space density (or compactness Q) before and after the deletion of that particular term, where Q is measured using the cosine similarity function. Thus, the the deletion of a good discriminators from the index vocabulary results in making the documents less distinguishable, i.e. high Q, whereas the deletion of bad discriminators makes the documents more distinguishable, that is low

Q. Can and Ozkarahan (1987) claim that there is an inverse relationship between Q and DCPL/N. That is, when documents are more distinguishable, Q is low, it also means that the documents are more decoupled from each other, and hence high DCPL/N On the other hand, when documents are less distinguishable, i.e. high Q, DCPL/N will be low.

To this end, they advocate the use of the value DCPL/N, or more specifically, the number of clusters , NC (by similar argument as in section 3.1, the constant N can be neglected), as an alternative way of calculating the discrimination value. Thus, the discrimination value for term i, can be computed using the number of clusters NC, and NC_i before and after deleting term i. i.e., $DV_i = NC - NC_i$.

From this definition, it follows that the deletion of good discriminators, while increasing the similarity of documents to each other, will decrease the number of clusters, i.e. $NC_i < NC$; Conversely, the removal of poor discriminators from the vocabulary, would decrease the similarity of documents and increase the number of clusters, i.e. $NC_i > NC$. Indifferent terms would not change the number of clusters, i.e. for such terms $NC_i = NC$.

The calculation of DV_i then requires the computation of NC and NC_i which are given by,

$$NC = \sum_{k=1}^{N} DCPL_k, \text{ and } NC_i = \sum_{k=1}^{N} DCPL_k^i$$

$$\text{where } DCPL_k = c_{kk} = \frac{1}{SUMROW_k} \sum_{j=1}^{M} \frac{d_{kj}^2}{SUMCOL_j}$$

$$\text{and } DCPL_k^i = c_{kk}^i = \frac{1}{SUMROW_k - d_{ki}} \sum_{stackrelj=1j\neq i}^{M} \frac{d_{kj}^2}{SUMCOL_j}$$

However, since the expressions for NC and NC_i differ only in the contribution made by term i, it is possible to formulate NC_i in terms of NC as

$$NC_i = \sum_{k=1}^{N} \frac{SUMROW_i DCPL_i - \frac{d_{ki}^2}{SUMCOL_i}}{SUMROW_i - d_{ki}}$$

Note that in this last expression, the summation is made over all of the documents, although most of the documents do not contain the deleted

term i. Thus, an alternative and relatively efficient formulation for NC may be obtained by modifying the values of NC only for those documents that contain the term i under consideration. In particular, for the documents indexed by i, NC can be computed by first eliminating the contribution of a document to the number of clusters (i.e. subtracting $DCPL_i$) and then reintroducing the contribution of the same document when the term is deleted. That is,

$$NC_i = NC + \sum_{k=1}^{NDI} \left(\frac{SUMROW_k DCPL_k - \frac{d_{ki}^2}{SUMCOL_i}}{SUMROW_k - d_{ki}} - DCPL_k \right)$$

where NDI refers to the number of documents indexed by term i. Accordingly, the expression for DV_i can be written as,

$$DV_i = NC - NC_i = \sum_{k=1}^{NDI} \left(DCPL_k - \frac{SUMROW_k DCPL_k - \frac{d_{ki}^2}{SUMCOL_i}}{SUMROW_k - d_{ki}} \right)$$

Figure 6 shows the algorithm for the foregoing procedure. It has been reported that the results obtained from this method are reasonably consistent with that obtained from the earlier methods based on the cosine coefficient.

A comparative evaluation of the computation involved in calculating discrimination values using the various similarity functions including Cover, Cosine, Dice, Overlap, and Jaccard coefficients is presented by Can and Ozkarahan (1987). It was noted that, as is evident from the algorithm, the discrimination value computation using the cover coefficient involves 6MN computations. Thus, the exact computation of discrimination values using the cover coefficient is much faster than that of algorithm D, while it compares favorably with that of algorithm C. It should, nevertheless, be noted that the cover coefficient technique does not exactly follow the idea of term discrimination and thus it is not possible to relate it as an alternative to any of the above procedures.

```
SUMSQC := 0;
FOR i := 1 TO M DO
  BEGIN
   c_i := 0;
   FOR j := 1 TO N DO
    C_i := c_i + d_ji;
   c_i := c_i / N
   SUMSQC := SUMSQC + c_i^2
  END;
Q := o;
FOR j := 1 TO N DO
  BEGIN
   SUMSQD_j := 0;
   DOTPROD_j := 0;
   FOR i := 1 TO M DO
    BEGIN
     SUMSQD_j := SUMSQD_j + d_ji^2;
     DOTPROD_j : DOTPROD_j + c_i d_ji;
    END
   Q = Q + cos(C, d_j)
  END
FOR i := 1 TO M DO
  BEGIN
   Q_i := Q;
   FOR j := 1 TO N DO
    IF d_ji > 0 THEN
       Q_i = Q_i - cos(C, d_j) + cos(C^i, d_j^i);
   DV_i = (Q_i - Q) / Q
  END.
```

Figure 3. Algorithm C

```
FOR i := 1 TO M DO
Q_i := 0;
Q := 0;
FOR j := 1 TO N - 1 DO
  FOR k := j + 1 TO N DO
    BEGIN
     Q := Q + cos(d_j, d_k);
     FOR i := 1 TO M DO
       IF d_ki > 0 THEN
         IF d_ji > 0 THEN
         Q_i := Q_i + DIFF1
         ELSE Q_i = Q_i + DIFF2
       ELSE
         IF d_ji > 0 THEN
         Q_i := Q_i + DIFF3
         ELSE SKIP
    END
  FOR i := 1 TO M DO
   DV_i := Q_i / Q;
```

Figure 4. Algorithm D

```
FOR i := 1 TO M DO
  BEGIN
    c_i := DIF_i := 0;
    FOR j := 1 TO N DO
      c_i := c_i + w_j d_ji;
  END;
FOR i := 1 TO M DO
  BEGIN
    FOR j := 1 TO N DO
      IF d_ji > 0 THEN
        FOR k := 1 TO M DO
          IF d_jk > 0 THEN
            DIF_k := DIF_k + (w_j^i - w_j) d_jk
    DV_i := 0;
    FOR k := 1 TO M DO
      DV_i := DV_i + DIF_k (DIF_k + 2c_k);
    DV_i := DV_i - c_i^2
END.
```

Figure 5. Algorithm E

```
FOR i := 1 TO N DO
BEGIN
  SUM := 0;
  FOR k := 1 TO M DO
    SUM := SUM + d²ik(SUMCOLk);
  DCPLi := SUM/SUMROWi
END
FOR i := 1 TO M DO
BEGIN
  DVi := 0;
  FOR j := 1 TO N DO
    IF dji > 0 THEN
      BEGIN
        SUMROWIj := SUMROWj - dji  j
        SUMCOLIi := d²ji/SUMCOLi
      DVi := DVi + (DCPLj - (DCPLjSUMROWj - SUMCOLIi)/(SUMROWIj))
      END
    ELSESKIP
END.
```

Figure 6. Algorithm F

Chapter 4

Experimental

4.1 Purpose

This study is concerned with the term discrimination value model advocated by Salton and his co-workers (Salton et al, 1976, Salton, 1975a) for term selection and term weighting. If we go back to their original idea, we recall that they argued that when documents are clustered, it is because they have been assigned similar sets of terms. This, they contend, is not a desirable situation for retrieval, since it is difficult to separate documents for retrieval. On the other hand, when the indexing leads to an even distribution of documents in the space, the separation of relevant from non-relevant documents is more difficult to achieve. The ideal situation is to cluster documents together so that the retrieval of similar (presumably relevant) documents and the rejection of dissimilar (non-relevant) documents is facilitated. In such a clustered document space, where the documents are grouped into classes on the basis of their relevance characteristics with respect to most user queries, Salton et al (1975a) claim that the best retrieval performance may be achieved through increasing similarity between documents in the same cluster (between documents with the same relevance characteristics), while decreasing the similarity between different clusters (i.e. between documents

with different relevance characteristics). In practice, this may be obtained by maximizing the average similarity between all pairs of documents within a single cluster, while at the same time minimizing the average similarity between different clusters.

The question then arises whether or not the current term discrimination values do test the foregoing. That is, whether or not the increase (decrease) in average similarity between all pairs of documents, regardless of their relevance characteristics, will worsen (improve) retrieval effectiveness. Salton et al (1975a) claim, on the basis of the experiments they carried out, that retrieval effectiveness is generally positively associated with separated (compressed) document spaces. In particular, assuming that any unclustered space may be approximated by the artificially created cluster space used in their experiment, and assuming that each cluster in their experiment normally exhibits identical relevance characteristics with respect to most user queries, they attempt to prove that the term discrimination value can be used to achieve the above mentioned situation. In simple terms, they claim that the term discrimination value rates terms according to their ability to effect a wider separation between relevant and non-relevant documents and a lesser separation among relevant documents. However, although superficially this attempt looks sensible enough, it suffers from the absence of relevance information and it is based on assumptions. Usually, in practice, documents may not fall into the said pattern of clusters; and the use of artificially created clusters may not always be effective, as it is liable to cause clusters to contain fairly dissimilar documents and at the same time similar documents may appear in different clusters. Therefore, it becomes clear from the foregoing that any attempt to deal with such question must take relevance information into account. However, such a test does not seem to have ever been carried out by workers in this area. The object of this study is, then, to set up experiments of this kind for collections with requests for which lists of relevant documents were available. More specifically, to test whether or not the discrimination value model leads to a distinction among

possible index terms in accordance with their ability to separate relevant documents from non-relevant documents. In the remainder of this chapter, the theory developed, and the experiments carried out are discussed. The discussion of the results obtained and conclusions reached are presented in the next chapter.

4.2 Theory

In order to test the foregoing, it is necessary to study the effect of each index term on the separation of relevant documents from each otheron the one hand and from non-relevant documents on the other. For this reason, this study involves a whole series of tests which have been designed to calculate and interprete the discriminating power of each term for the relevant-relevant distribution (DVR), relevant - non-relevant document distribution (DVRNR), and also for the non-relevant - non-relevant document distribution (DVNRNR).

The calculation of DVR, DVRNR, and DVNRNR takes as its basis some of the methods discussed in the previous chapter, in particular the algorithm proposed by EL-Hamdouchi and Willett (1988a).

As mentioned earlier, the calculation of DV values involves the sum of all the distinct inter-document similarities,

$$Q = \sum_{\substack{j,k=1 \\ j<k}}^{N} w_j w_k DOTPRODJK, \quad and$$

$$Q_i = \sum_{\substack{j,k=1 \\ j<k}}^{N} w_j^i w_k^i DOTPRODJK$$

$$where \quad DV_i = Q_i - Q$$

The relevance judgement for a given query divides the document collection into two sets; namely, the set of relevant documents and the set of non-relevant documents. Therefore, the sum of the distinct inter-document

similarities, Q, can be reduced to the sum of (i) the sum of the distinct inter-document similarities taken over all pairs of documents both of which are relevant (QR), (ii) the sum of the inter-document similarities taken over all pairs of documents one of which is relevant and the other non-relevant (QRNR), and (iii) the sum of the inter-document similarity taken over all pairs of documents both of which are non-relevant (QNR). That is,

$$Q = QR + QNR + QRNR$$

$$\textit{where,} \quad QR = \sum_{\substack{j,k=1 \\ j<k}}^{NMR} w_j w_k DOTPRODJK,$$

$$QNR = \sum_{\substack{j,k=1 \\ j<k}}^{NMNR} w_j w_k DOTPRODJK,$$

$$QRNR = \sum_{j=1}^{NMR} \sum_{k=1}^{NMNR} w_j w_k DOTPRODJK$$

where N,NMR,NMNR refer to total number of documents, number of relevant documents, and number of non-relevant documents respectively. Similarly,

$$QI = QRI + QNRI + QRNRI$$

$$\textit{where,} \quad QRI = \sum_{\substack{j,k=1 \\ j<k}}^{NMR} w_j^i w_k^i DOTPRODJK,$$

$$QNRI = \sum_{\substack{j,k=1 \\ j<k}}^{NMNR} w_j^i w_k^i DOTPRODJK,$$

$$QRNRI = \sum_{j=1}^{NMR} \sum_{k=1}^{NMNR} w_j^i w_k^i DOTPRODJK$$

Therefore, by substitution, $DV_i = Q_i - Q$ may be formulated as:

$$\begin{aligned} DV_i &= (QRI + QNRI + QRNRI) - (QR + QNR + QRNR) \\ &= (QRI - QR) + (QNRI - QNR) + (QRNRI - QRNR) \\ &= DVR_i + DVNR_i + DVRNR_i \end{aligned} \tag{4.1}$$

by the definition of DV' where $DVR_i, DVNR_i, DVRNR_i$ refer to the DV_i values with respect to the relevant - relevant, non-relevant - non-relevant, and relevant - non-relevant document.

Hence, for a given query, the calculation of DVR involves the selection of the relevant documents and the corresponding index terms used, and computation of DVR values using one of the methods described above. Due to its remarkable efficiency the algorithm proposed by El-Hamdouchi and Willett is used in this study. However, rather than using this algorithm for the calculation of the DV values for terms in the whole collection, DVR values may be obtained simply by running the algorithm on that small subset of the database comprising the relevant documents for some query. Similarly, the DVNR values may be calculated using the non-relevant subset. In the case of DVRNR , it is not possible to use the algorithm directly, since we are here dealing with two sets of documents. However, it can be obtained by first calculating DV, DVR and DVNR and then evaluating DVRNR from

$$DVRNR_i = DV_i - DVR_i - DVNR_i.$$

4.3 Calculation of DVRNR Values

The above procedure can be applied to any form of document collection with relevance judgement which lends itself to the computation of DV values. However, the main practical difficulty is the computation involved, in particular the need to calculate DVNR and DV values as a precursor to the evaluation of the DVRNR values; it may be, of course, that these values are also required (which is the case in our experiment). When one needs to calculate the DVRNR values alone, the following approach may be considered.

Consider the equation

$$SUMSQC = \sum_{j,k=1}^{N} w_j w_k DOTPRODJK$$

from the previous chapter. This can be formulated as:

$$SUMSQC = 2 \sum_{\substack{j,k=1 \\ j<k}}^{N} w_j w_k DOTPRODJK + \sum_{j=1}^{N} w_j^2 SUMSQD_j \quad (4.2)$$

from which Q may be obtained as:

$$2Q = SUMSQC - \sum_{j=1}^{N} w_j^2 SUMSQD_j$$

With this consideration in mind, let us once again examine the sum of the inter-document similarity between all pairs of documents.

$$\begin{aligned} SUMSQC &= \sum_{j,k=1}^{N} w_j w_k DOTPRODJK \\ &= 2QR + 2QNR + 2QRNR + \sum_{j}^{N} w_j^2 SUMSQD_j \end{aligned}$$

where QR, QNR, and QRNR are as defined earlier. Rearranging this last equation we get:

$$2QRNR = SUMSQC - 2QR - 2QNR - \sum_{j=1}^{N} w_j^2 SUMSQD_j$$

After applying (1.2) above on the set of relevant documents, and on the set of non-relevant documents, we obtain:

$$\begin{aligned} 2QRNR = SUMSQC &- \left(SUMSQCR - \sum_{j=1}^{NMR} w_j^2 SUMSQD_j \right) \\ &- \left(SUMSQCNR - \sum_{j=1}^{NMNR} w_j^2 SUMSQD_j \right) \\ &- \sum_{j=1}^{N} w_j^2 SUMSQD_j \end{aligned}$$

Where SUMSQCR and SUMSQCNR refer to the sum of the squares of the elements in the centroid of relevant documents, and the sum of the squares of the elements in the centroid of the non-relevant documents respectively.

Note that, for a given collection,

$$\sum_{j=1}^{N} w_j^2 SUMSQD_j = \sum_{j=1}^{NMR} w_j^2 SUMSQD_j + \sum_{j=1}^{NMNR} w_j^2 SUMSQD_j$$

Therefore, after the obvious simplification of the above, we get:

$$2QRNR = SUMSQC - SUMSQCR - SUMSQCNR$$

Accordingly,

$$2QRNRI = SUMSQCI - SUMSQCRI - SUMSQCNRI$$

Where SUMSQCRI , SUMSQCNRI refer to SUMSQCR, SUMSQCNR respectively, when the i-th term is deleted.

Therefore, the effect of an individual term i upon the separation of relevant documents from non-relevant documents may be determined by,

$$\begin{aligned} DVRNR_i &= QRNRI - QRNR \\ &= (SUMSQCI - SUMSQCRI - SUMSQCNRI) \\ &- (SUMSQC - SUMSQCR - SUMSQCNR) \end{aligned}$$

Obviously rearranging this last equation one can obtain (eqn. 1.1) above. However, if we take into account the relationship between the centroids; i.e., C = CR + CNR (where CR and CNR refer to the centroid of relevant and non-relevant documents respectively), SUMSQC may be written as,

$$SUMSQC = SUMSQCR + SUMSQCNR + 2DOTPRODCRCNR,$$

where DOTPRODCRCNR refers to the dot product of CR and CNR. Similarly,

$$SUMSQCI = SUMSQCRI + SUMSQCNRI + 2DOTPRODCRICNRI.$$

Substituting these in the above we get,

$$\begin{aligned} DVRNR_i &= 2DOTPRODCRICNRI - 2DOTPRODCRCNR \\ &= \sum_{k=1}^{M} CRI_k CNRI_i - \sum_{k=1}^{M} CR_k CNR_k \\ &= \sum_{k=1}^{M} (CRI_k CNRI_k - CR_k CNR_k) \end{aligned} \qquad (4.3)$$

where M refers to the total number of possible indexing terms.

The important thing to note about this last expression is that, unlike (4.1) above, it involves only the calculation of CRI, CNRI and the sum of

their product for the calculation of DVRNR , since for a given query CR, and CNR are constant. Thus, DVRNR may be evaluated simply by calculating the centroids of the sets of relevant and non-relevant documents when the i-th term is, and is not, used for indexing. However, in the experiments reported, since we were also interested in the DVNR values, we used (4.1) above. It should nevertheless be mentioned that the equivalence of the two methods, for calculating DVRNR, (4.1) and (4.3), was proved experimentally.

4.4 Method

The experiments followed closely the theoretical approach discussed above in section 1.2. In order to calculate DVR , first similarities between all pairs of documents both of which are relevant to a given query are computed, and these are summed to produce the overall similarity of the relevant documents. A new similarity value is then calculated by removing term i from the set of index terms. DVR is then obtained as the difference of these two similarities. Having found the DVR values, the ranks of all of the terms appearing at a particular frequency in the relevant documents (in the whole collection) are then averaged and a graph is produced against frequency within the relevant documents (against collection frequency).

Similarly, for DVRNR , the overall similarity of relevant - non-relevant documents is obtained by summing the similarities between all possible pairs of documents one of which is relevant and the other non-relevant. A new overall similarity is then calculated by deleting the term from the set of index terms. The difference of these two similarities will then give DVRNR . After calculating DVRNR for all terms, they are ranked and plotted against collection frequency. In this case, it was also felt helpful to produce, (i) the same graph but for those terms which only appear in relevant documents, (ii) the same graph but for those terms which appear both in relevant and

non-relevant documents, and (iii) the same graph but for those terms which only appear in non-relevant documents. Finally, to study the effect of this technique on the separation of non-relevant from non-relevant documents, the corresponding discrimination values (DVNRNR) were calculated in a similar fashion, ranked, and plotted against collection frequency. For further implementation details, please see the program coding (Appendix B).

The foregoing procedure may be performed for every query in a given collection. However, to be able to generalize the results, it was felt sufficient to consider only those queries with more than 25 relevant documents in each collection.

4.5 Data

The experiments discussed in the subsequent section of this chapter used five different document collections. These are the KEEN collection, the EVANS collection, the HARDING collection, the CRANFIELD 1400 collection, and the LISA collection. These collections were used because they are from different sources, and are large enough to ensure statistically reliable results. While their general characteristics are similar (see Table 4.1 below) they have markedly different collection characteristics (Griffiths et al, 1986)

In order to test the validity of the result, it was felt that it would be of benefit to use such different document collections for the sake of comparison. If the results are similar for all of the collections it could then be argued that they are not collection dependent and will thus hold for any document set. A further point to note is that these collections may be used by other research projects, thereby allowing comparisons between different workers.

All the collections consist of a set of documents and queries expressed in terms of the indexing terms used to describe them, and a set of relevance judgements associated with each query. The documents are represented

in terms of binary vectors. That is, the only information available states whether a term is used to index the document or not, a 0 if the term is not used, and a 1 otherwise. Such a representation simplifies many of the calculations; thus the vector product in the cosine coefficient is simply the number of terms in common, while the sum of squares of the elements in the document vectors is just the document length (the number of indexing terms used to describe a document).

The documents were stored in our computer implementation as a series of numbers, the first indicating the number of the document in the sequence, the second being the document length, and the rest being the indexing terms used to describe the document, these being arranged in numerically ascending order. For the indexing terms, the inverted file to the document collection is used. It consists of a set of lists of document numbers, where each list, for each term, consists of a series of numbers, the first indicating the term number, the second being the number of documents indexed by it, and the rest being the document numbers. The same holds for the relevance judgement file, i.e., the first number indicates the query number, then the number of relevant documents followed by the actual document numbers. All these arrangements minimize the amount of storage required since only the non-zero elements in the representative vector are stored.

The table below summarizes some of the details of these collections; further details may be found in (Griffiths et al, 1986).

	Keen	Cranfield	Evans	Harding	LISA
Number of documents	800	1400	2542	2472	6004
Number of queries	63	225	39	65	35
Number of index terms	1432	2557	3730	8783	13355
Mean number of terms per doc.	9.8	28.7	6.6	36.3	39.7
Mean number of terms per query	10.3	8.0	27.5	32.4	16.5

Table 4.1: Characteristics of the document test collection

Chapter 5

Results

5.1 Discussion of Results

The results obtained from this study are embodied in the various graphs presented in appendix A. These are conveniently organised in eight sets, where each set contains five graphs, one from each collection. They are also arranged to allow convenience in discussing them under the four major cases, that is, the case for the entire collection, the relevant-relevant document distribution, the non-relevant - non-relevant document distribution, and the relevant - non-relevant document distribution. All the graphs, except the ones in the first set, represent selections from the complete set of results of the experiments that have been carried out. As has been indicated in the preceding chapter, all queries with more than 25 relevant documents were tested in this study. However, since similar results were obtained from different queries within the same collection, it was felt sufficient to consider the results of an arbitrary query in each collection, for the purpose of analysis and presentation.

The first set (Graphs I.1 - 5) shows the results obtained when the algorithm proposed by El-Hamdouchi and Willett (1988a) is used to calculate the term discrimination values, for the entire collection. As may be seen

from the plots, a U shaped curve is obtained for all the collections, which suggests that the medium frequency terms are the best discriminators. Thus it is possible, using this new algorithm, to reproduce the results obtained in previous experiments (Salton et al, 1975a, Willett, 1985).

The next two sets of graphs relate to the discrimination characteristics of indexing terms in the space of relevant documents. The results in the first of these two sets (II.1 - 5) are indeed quite interesting as they were not what one might expect on the basis of the discussion in section 3.2. It has long been argued (see chapter 3) that medium frequency terms, that is, terms whose document frequency is neither too low or too high, are the best discriminators and that their use will lead to the best retrieval performance. On the other hand, the best retrieval performance is associated with a wide separation of relevant from non-relevant and the close association of relevant documents to each other. Therefore, it is to be expected that the medium frequency terms should effect a lesser separation of the relevant documents than do the infrequent and frequent terms. However, as may be seen from the graphs, the low and high frequency terms do better than the medium frequency terms in effecting a smaller separation of relevant documents from each other; in fact, the high frequency terms seem to do best of all of the terms.

In order to generalize this observation for the entire collection, it is necessary to interprete these results in terms of the entire collection frequency. It has been noted (Yu et al, 1982) that low frequency terms used for indexing purposes tend to be more concentrated in relevant documents, and thus it is convenient to assume high frequency terms in relevant documents to be low frequency terms in the entire collection. In the light of this and on the basis of the results obtained in this study, one may conjecture that generally low frequency terms (that is, high frequency terms in the relevant documents) separate relevant documents less from each other than medium frequency terms. In an attempt to support this point experimentally, the average rank

(for DVR) was plotted against collection frequency (as shown in set III). These graphs are highly unstructured and it is not possible to identify any obvious relationship between DVR and frequency. Taken together, the results in these two sets of graphs (II and III) would seem to contradict the assertion that medium frequency terms are the best index terms.

On the other hand, as far as the space of relevant documents is concerned (i.e, as a collection in its own right), these results are in accord with the results obtained in previous work with the exception of the EVANS collection where the low frequency terms appear to be the best discriminators (this is, may be, because the EVANS collection has rather different collection characteristic (El-Hamdouchi and Willett, 1988b); for instance, it contains very long queries which contain an average of 27.5 indexing terms). Therefore, in this circumstances, the assertion that medium frequency terms are best discriminators may be modified to include low frequency terms, that is, not only medium but low to medium frequency terms are best discriminators.

The fourth set of graphs shows the results obtained from calculating the DVNRNR values for the non-relevant documents. The graphs seem to be fairly similar to those in the first set, and thus confirming the assumption repeatedly made in probabilistic indexing; i.e.; the assumption that the characteristics of the non-relevant documents can be approximated by those of the entire collection (Rijsbergen, 1979).

The next sets of plots relate to the case of relevant - non-relevant document discrimination. The first of these, Graph V, shows the characteristics of all the terms in discriminating the relevant documents from the non-relevant ones. The graphs demonstrate, in most cases, that the medium to high frequency terms exhibit the mixture of both good and bad discriminators. In order to investigate this in greater detail, the subsequent sets of graphs (VI-VIII) were produced, that is, the same graph but for those terms which only appear in relevant documents, the same graph but for those terms which appear in both, and the same graph but for those terms

which only appear in non-relevant respectively. For terms which appear only in relevant or only in non-relevant (refer to set VI and set VIII), there is no longer a minimum rank (that is, high discrimination value) for the medium frequency terms, instead the best discriminators, i.e. those having the lowest ranks, are those occurring with high frequencies. In fact, Graph VI suggests that the best discriminators which appear only in relevant documents are those terms with frequency 1 (in most cases), an observation which is in agreement to the IDF system. As far as the terms in both are concerned (set VII), whilst low frequency terms exhibit relatively average discriminators (not very good and not very bad), once again medium frequency terms include both the best and worst discriminators. In most cases high frequency terms include good and bad discriminators. Thus, from the relationships demonstrated by these graphs, it follows that high frequency terms in relevant and non-relevant documents are very good separators of relevant documents from non-relevant ones. On the basis of these graphs, one may also conjecture the possible inclusion of good discriminators in high and low frequency terms, which is also supported by the results in set 3; an observation in accord with that of Salton et al (1975a). In addition, the plots demonstrate that many bad discriminators are included in the set of medium frequency terms (which is again supported by the results in set 3).

5.2 Conclusion

The term discrimination model as a term selection and term weighting method has been intensively studied ever since the early 1970s, when the idea was first presented. The basic assumption in the term discrimination theory is that a greater separation between documents will enhance retrieval effectiveness but that a lesser separation will depress the retrieval. However, what is really required is to separate relevant documents from non-relevant documents, while at the same time moving relevant documents close to each other. This study, was, therefore, set out to test whether or not the use of

the term discrimination model as an indexing strategy leads to this situation. To this end an extensive series of tests has been carried out using the KEEN, CRANFIELD, EVANS, HARDING, and LISA test collections. As to the computation of the discrimination values, the algorithm proposed by El-Hamdouchi and Willett (1988a) is used. In addition, an efficient way of computing exact term discrimination values for relevant - non-relevant document distribution is introduced which involves only the use of the centroids of the relevant documents and non-relevant documents.

The results of this study show that good index terms, as far as the relevant documents separation is concerned, are high frequency terms within the relevant documents. The index terms that best separate between relevant and non-relevant documents are high frequency terms which appear either only in relevant documents or only in non-relevant documents. The results also suggest that, contrary to previous results, medium frequency terms do contain both best and worst separators of relevant documents from each other on the one hand and from non-relevant documents on the other.

Taken together, as these results are common among the collections tested here, it can be concluded that independent of the collection, medium frequency terms are not the best index terms in the sense that they do not always minimise the separation among relevant documents and maximise the separation between relevant and non-relevant documents. Hence, the term discrimination value rates index terms not on their ability to separate relevant documents from non-relevant ones, but on their ability to separate documents from each other without taking account of their relevance characteristics.

Bibliography

[1] Adamson G. W. and Boreham J. (1974), The use of an association measure based on character structure to identify semantically related pairs of words and document titles, *Information Storage and Retrieval* , **10** , pp 253-260.

[2] Booth A. D. (1967), A law of occurrence for words of low frequency, *Information and Control* , **10** (4), pp 386-393.

[3] Borko H. (1977), Towards a theory of indexing *Information Processing and Management*, **13**(6), pp 355-365.

[4] Bratley P. and Choueka Y. (1982), Processing truncated terms in document retrieval systems, *Information Processing and Management*, **18**, pp 257-266.

[5] Can F. and Ozkarahan E. A. (1987), Computation of term/document discrimination values by use of the Cover Coefficient concept, *Journal of the American Society for Information Science*, **38** (3), pp 171-183.

[6] Can F. and Ozkarahan E. A. (1984), Two patitioning type clustering algorithms, *Journal of the American Society for Information Science*, **35** (5), pp 268-276.

[7] Cleverdon C. W. (1967), The Cranfield tests on indexing language devices, **Aslib Proceedings, 16** (6), pp 173-194.

[8] Crawford R. G. (1975), The Computation of discrimination values, *Information Processing and Management*, **11** (8/12), pp 249-253.

[9] Croft W. B. and Harper D. J. (1979), Probabilistic models of document retrieval without relevance information, *Journal of Documentation*, **35** (4), pp 285-295.

[10] Croft W. B. (1981), Document representation in probabilistic models of information retrieval, *Journal of the American Society for Information Science*, **32**, pp 451-457.

[11] Croft W. B. (1983), Experiments with representation in a document retrieval system, *Information Technology: Research and Development*, **2**, pp 1-21.

[12] Croft W. B. and Harper D. J. (1979), Using probabilistic models of document retrieval without relevance information, *Journal of documentation*, **35**, pp 285-295.

[13] El-Hamdouchi A. and Willett P. (1988a), An improved algorithm for the calculation of exact term discrimination values, *Computer Journal.*

[14] El-Hamdouchi A. and Willett P. (1988b), Technique for the measurement of clustering tendency in document retrieval system, *Journal of Information Science.*

[15] Frakes W. B. (1984), Term conflation for information retrieval, In: C. J. van Rijsbergen (ed), Research and Development in Information Retrieval, Cambridge: CUP, pp 383-390.

[16] Freund G. E. and Willett P. (1982), Online identification of word identification of word variants and arbitrary truncation searching using a string similarity measure, *Information Technology: Research and Development*, **1**, pp 177-187.

[17] Griffiths A., Luckhurst H. C. and Willett P. (1986), Using interdocument similarity information in document retrieval systems, *Journal of the American Society for Information Science*, **37** (1), pp 3-11.

[18] Harper D. J. and Rijsbergen C. J. (1978), An evaluation of feedback in document retrieval using co-occurrence data, *Journal of Documentation* , **34** , pp 189-216.

[19] Harter S. P. (1975), A probabilistic approach to automatic keyword indexing, Part I: On the distribution of speciality words in technical literature. Part II: An algorithm for probabilistic indexing, *Journal of the American Society for Information Science* , **26** , pp 197-206, 280-289.

[20] Harter S. P. (1978), Statistical approach to automatic indexing, *Drexel Library Quarterly*, **14**, pp 57-74.

[21] Hendry I. G. (1983), A consideration of the term discrimination value as a method for automatic indexing, *M.SC. thesis* , University of Sheffield.

[22] Lennon M., Peirce D. S., Tarry B. D. and Willett P. (1981), An evaluation of some conflation algorithms for information retrieval, *Journal of Information Science*, **3**, pp 177-183.

[23] Lovins J. B. (1968), Development of a stemming algorithm, *Mechanical Translation and Computational Linguistics*, **11**, pp 22-31.

[24] Luhn H. P. (1957), A statistical approach to mechanized encoding and searching of library information, *IBM Journal of Research and Development* , **1** , pp 309-317.

[25] Montgomery C. A. (1972), Linguistics and information science, *Journal of the American Society for Information Science*, **23**, pp 195-219.

[26] Porter M. F. (1980), An algorithm for suffix stripping, *Program*, **14**, pp 130-137.

[27] Porter M. F. (1982), Implementing a probabilistic retrieval system, *Information Technology: Research and Development*, **1**, pp 131-156.

[28] Robertson S. E. (1974), Specificity and weighting, *Journal of Documentation*, **30**, pp 41-46.

[29] Robertson S. E. (1977a), Theories and Models in information retrieval, *Journal of Documentation*, **33** , pp 126-148.

[30] Robertson S. E. (1977b), The probability ranking principle in information retrieval, *Journal of Documentation*, **42**, pp 182-188.

[31] Robertson S. E. (1986), On relevance weight estimation and query expansion, *Journal of Documentation*, **42**, pp 182-188.

[32] Robertson S. E. and Sparck Jones K. (1976), Relevance weighting of search term, *Journal of the American Society for Information Science*, **27**, pp 129-146.

[33] Rowbottom M. E. and Willett P. (1982), The effect of subject matter on the automatic indexing of full text, *Journal of the American Society for Information Science*, May, pp 139-141.

[34] Salton G. (1971), The SMART Retrieval System, Englewood Cliffs: Prentice-Hall.

[35] Salton G. (1975a), A theory of indexing, Regional conference series in applied mathematics, no. 18, Philadelphia : Society for Industrial and Applied Mathematics.

[36] Salton G. (1975b), Dynamic information and library processing, Englewood Cliffs, New Jersey: Prentice-Hall.

[37] Salton G. (1986), Recent trends in automatic information retrieval, *Proceeding of the Ninth International Conference on Research and Development in Information Retrieval* , Washington: ACM, pp 1-10.

[38] Salton G. and McGill M. J. (1983), Introduction to modern information retrieval, New York, London : McGraw-Hill.

[39] Salton G. and Wong A. (1976), On the role of words and phrases in automatic text analysis, *Computers and Humanities*, **10** (2), pp 69-87.

[40] Salton G., Wong A. and Yang C. S. (1975a), A vector space model for automatic indexing, *Communications of the ACM*, **18** (11), pp 613-620.

[41] Salton G., Wong A. and Yu C. T. (1976), Automatic indexing using term discrimination and term precision measurements, *Information Processing and Management* , **12** (1), pp 43-51.

[42] Salton G. and Wu H. (1981), A term weighting model based on utility theory, In Oddy R. N. et al, Information retrieval research, Landon: Butterworths, pp 9-22.

[43] Salton G., Wu H. and Yu C. T. (1981), The measurement of term importance in automatic indexing, *Journal of the American Society for Information Science*, **32** (3), pp 175-186.

[44] Salton G. and Yang C. S. (1973), On the specification of term values in automatic indexing, *Journal of Documentation* , **29** (4), pp 351-372.

[45] Salton G., Yang. C. S. and Yu C. T. (1975b), A theory of term importance in automatic text analysis, *Journal of the American Society for Information Science* , **26** (1), pp 33-44.

[46] Sparck Jones K. (1971), Automatic Keyword Classification for Information Retrieval, London: Butterworth.

[47] Sparck Jones K. (1972), A statistical interpretation of term specificity and its application in retrieval, *Journal of Documentation* , **28** (1), pp 11-21.

[48] Sparck Jones K. (1973), Index term weighting, Information Storage and Retrieval, 9 (11), pp 619-633.

[49] Sparck Jones K. (1974), Automatic indexing, *Journal of Documentation* , **30**, pp 393-432.

[50] Sparck Jones K. (1979a), Search term relevance weighting given little relevance information, *Journal of Documentation*, **35**, pp 30-48.

[51] Sparck Jones K. (1979b), Experiments in relevance weighting of search terms, *Information Processing and Management*, **15**, pp 133-144.

[52] Sparck Jones K. (1982), Information Retrieval Experiment, London: Butterworth.

[53] Sparck Jones K. and Bates R. G. (1977), Research on automatic indexing 1974-7976, Cambridge : the University Computer Laboratory.

[54] Sparck Jones K. and Kay M. (1973), Linguistics and Information Science, Academic Press, New York, and London.

[55] Sparck Jones K. and Tait J. I. (1984), Automatic search term variant generation, *Journal of Documentation*, **40**, pp 50-66.

[56] Trivision D. (1987), Term co-occurrences in cited/citing journal articles as a measure of document similarity, *Information Processing and Management* , **23** (3), pp 183-194.

[57] Ulmschneider J. E. and Doszkocs T. (1983), A practical stemming algorithm for online search assistance, *Online Review*, **7** , pp 301-318.

[58] Van Rijsbergen C. J. (1977), A theoretical basis for the use of co-occurrence data in information retrieval, *Journal of Documentation* , **33** , pp 106-119.

[59] Van Rijsbergen C. J. (1979), Information Retrieval, London: Butterworth.

[60] Voorhees E. (1986), Implementing agglomerative hierarchic clustering algorithms for use in document retrieval, *Information Processing and Management* , **22**, pp 465

[61] Willett P. (1981), A fast procedure for the calculation of similarity coefficients in automatic classification, *Information Processing and Management* , **17** (2), pp 53-60.

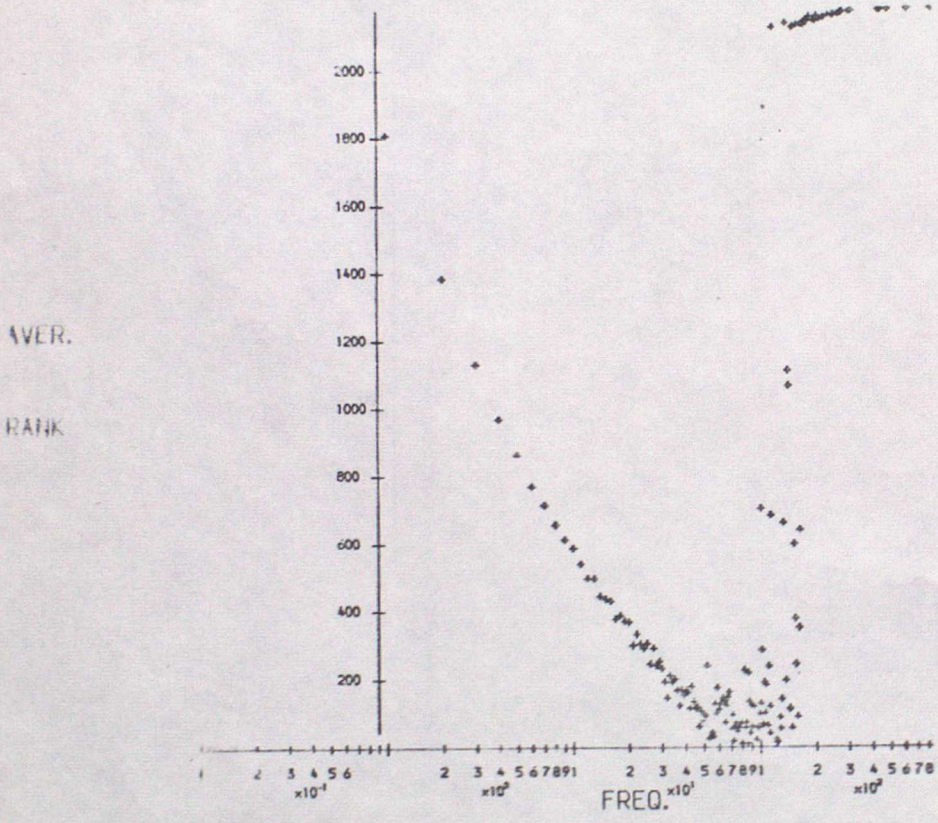
[62] Willett P. (1985), An algorithm for the calculation of exact term discrimination values, *Information Processing and Management* , *21*, pp 225-232.

[63] Wu H. and Salton G. (1981a), A comparison of search term weighting: Term Relevance vs. Inverse Document Frequency, SIGIR Forum, 1981, 16, pp 30-39.

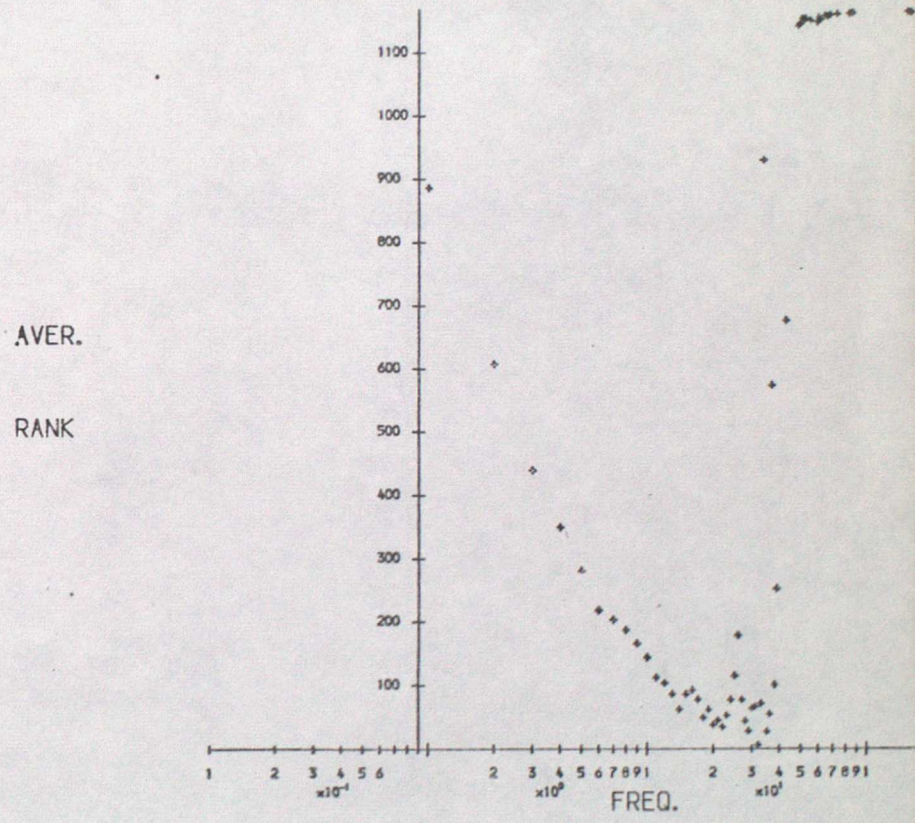
[64] Wu H. and Salton G. (1981b), The estimation of term relevance weights using relevance feedback, *Journal of Documentation* , 37 (4), pp. 194-214.

[65] Yu C. T., Lam K. and Salton G. (1982), Term weighting in information retrieval using the term precision model, *Journal of the Association for computing machinery*, 29 (1), pp 152-170.

[66] Yu C. T. and Salton G. (1977), Effective information retrieval using term accuracy, *Communication of the ACM* , 20 (3), pp 135-142.



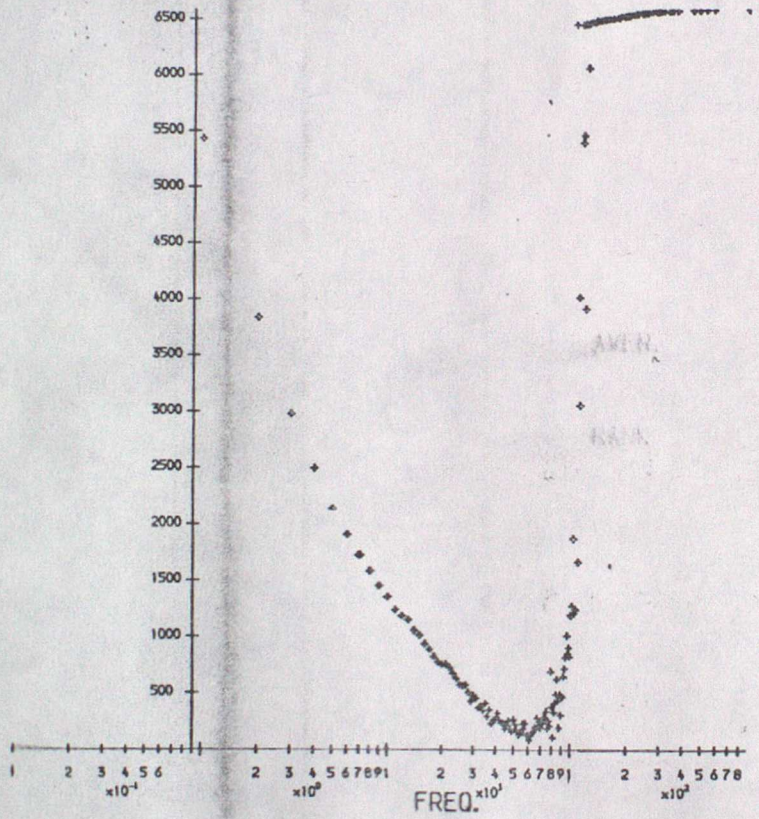
GRAPH I. 2. AVER. (DV) RANK VS COLL. FREQ. PLOT FOR THE CRAN COLLECTION



GRAPH I. 1. AVER. (DV) RANK VS COLL. FREQ. PLOT FOR THE KEEN COLLECTION

AVER.

RANK

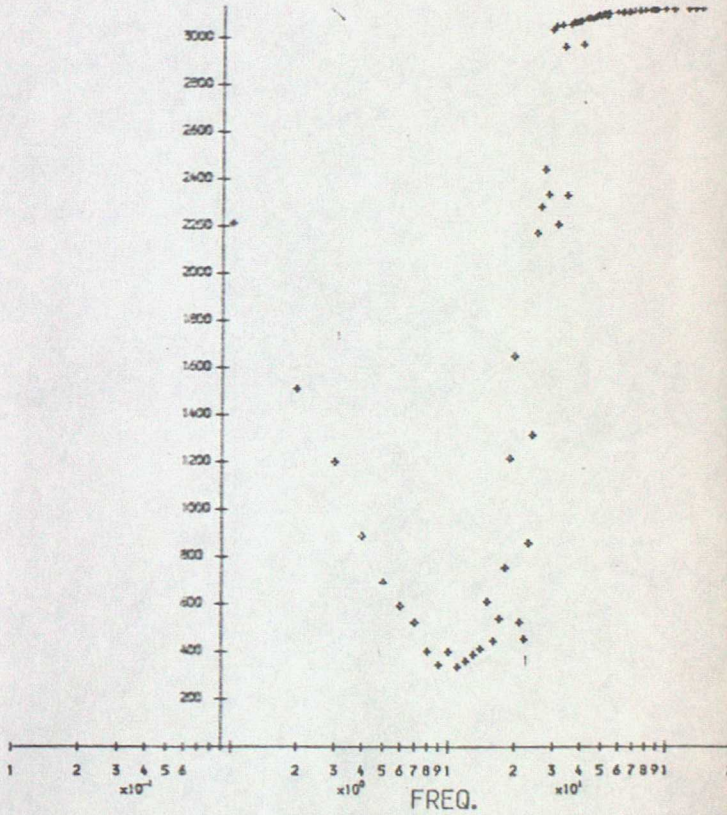


GRAPH I. 4. AVER. (DV) RANK VS COLL. FREQ. PLOT FOR THE HARD COLLECTION

A-4

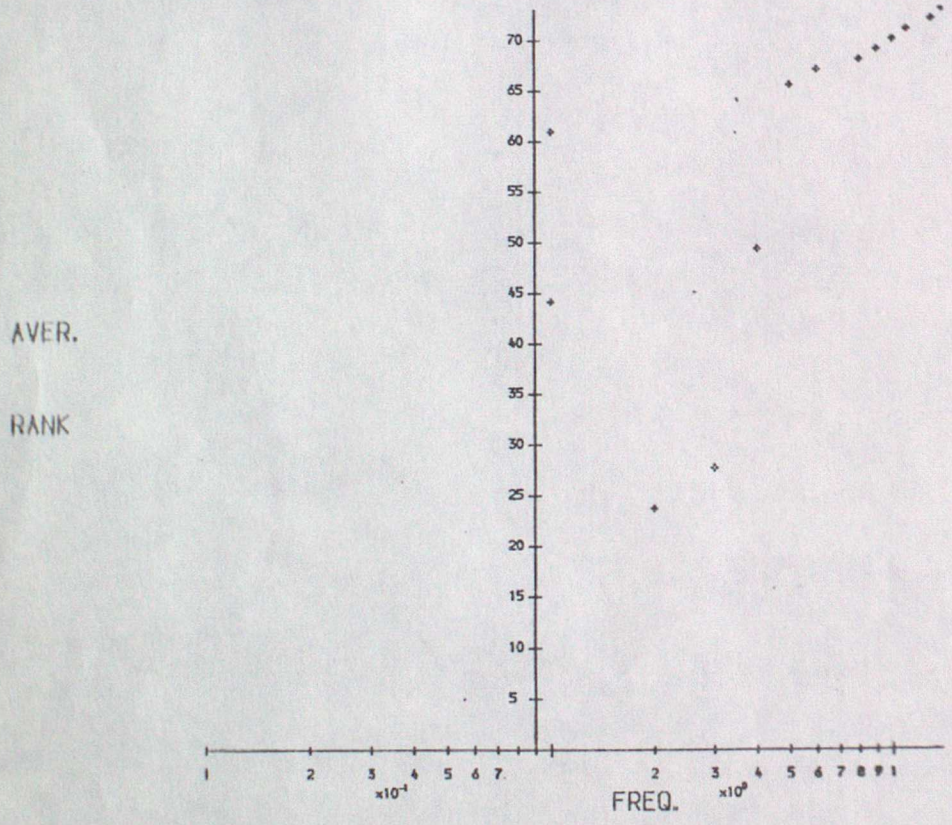
AVER.

RANK

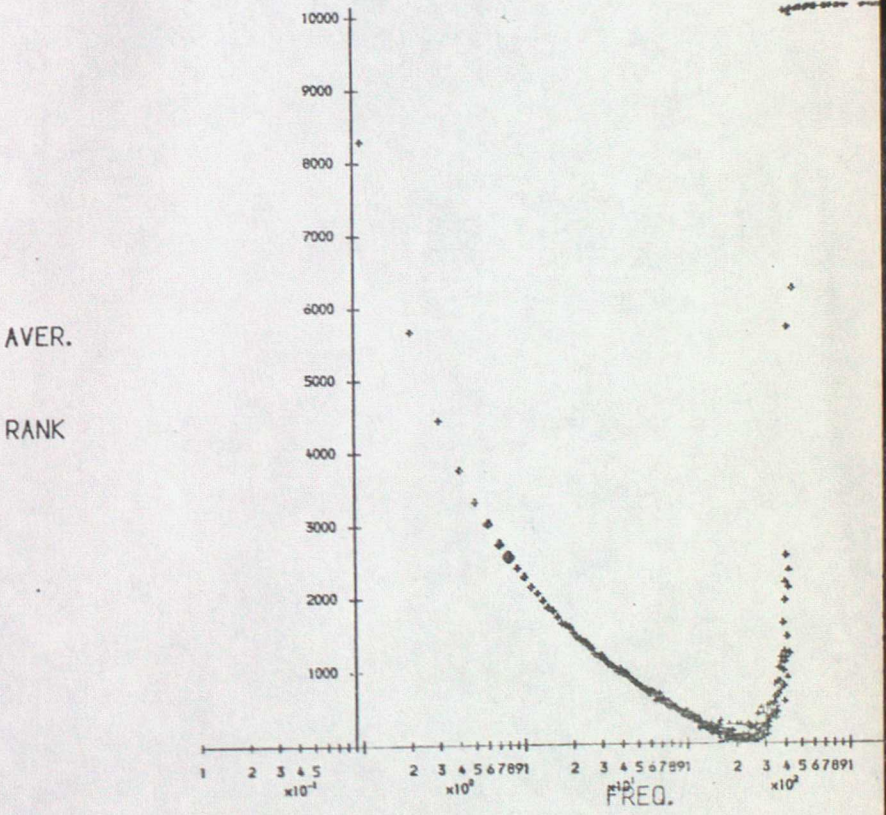


GRAPH I. 3. AVER. (DV) RANK VS COLL. FREQ. PLOT FOR THE EVAN COLLECTION

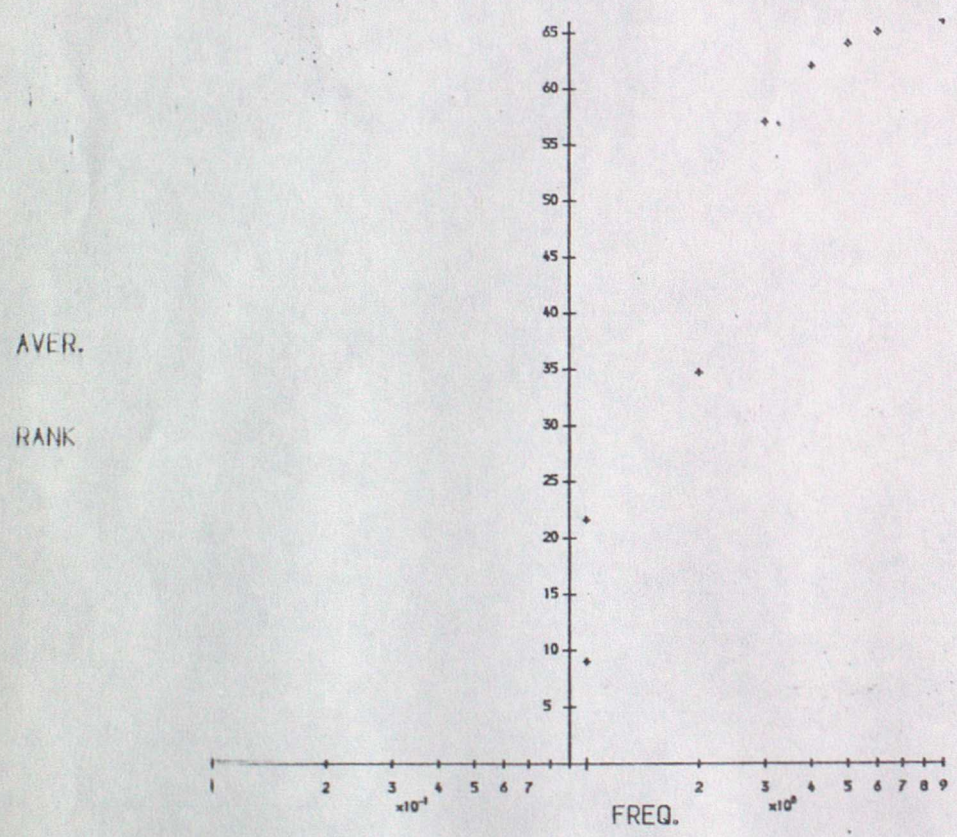
A-3



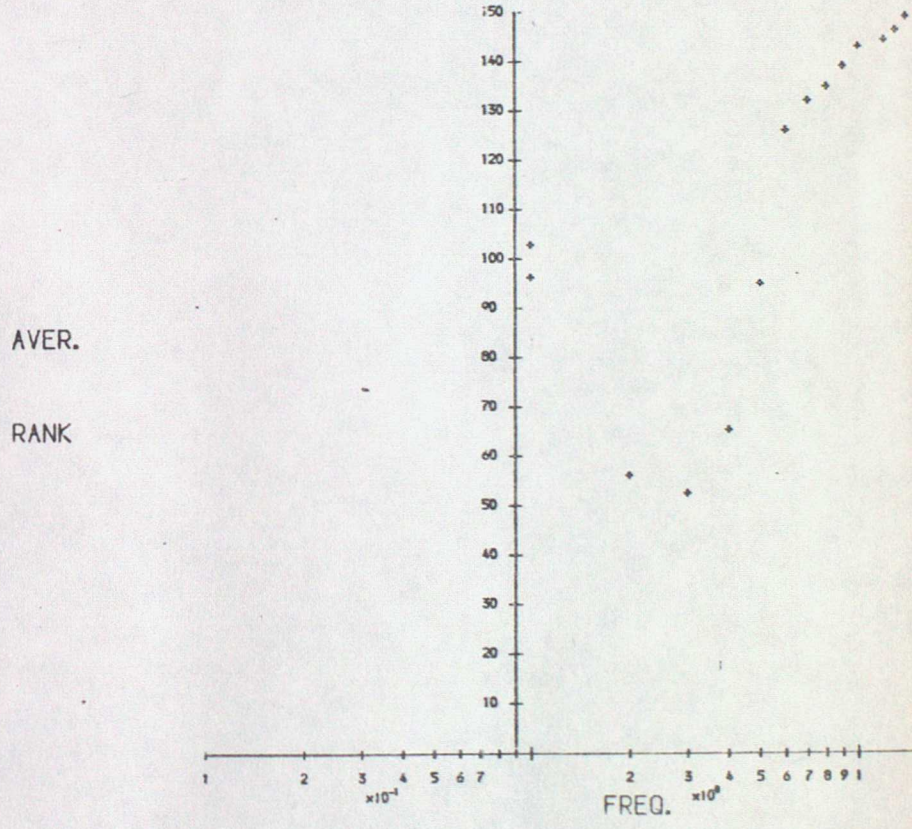
GRAPH II. 1 : (DVR) RANK VS FREQ. WITHIN REL. DOC. PLOT FOR THE KEEN COLLECTION



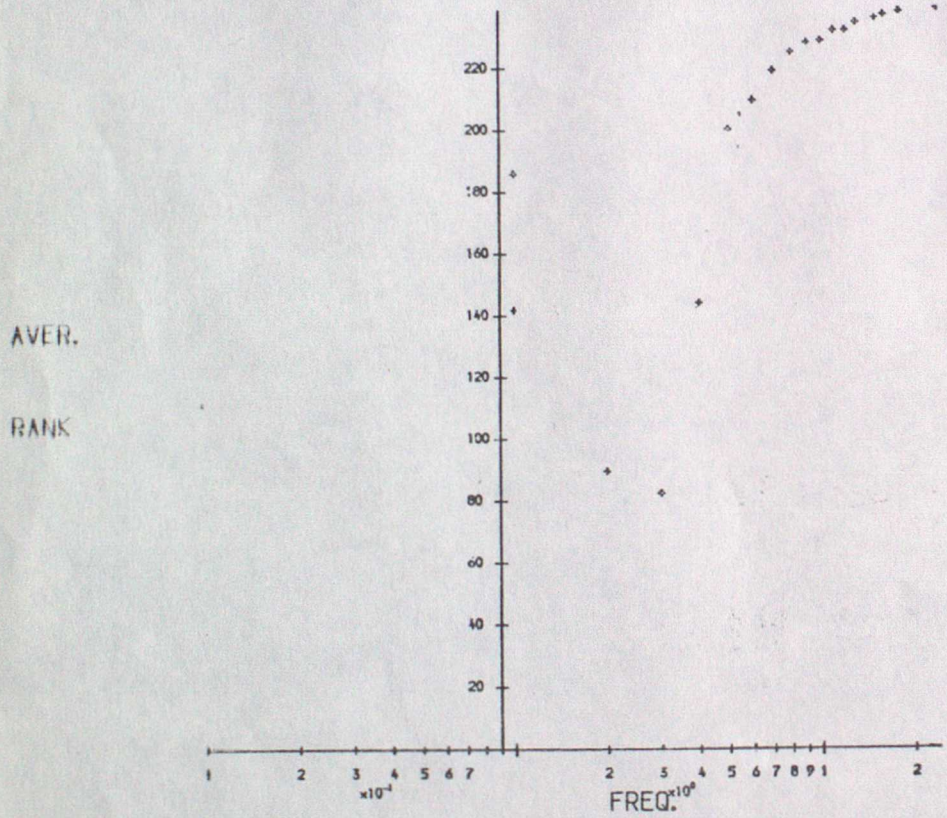
GRAPH I. 5₂ AVER. (DV) RANK VS COLL. FREQ. PLOT FOR T
LISA COLLECTION



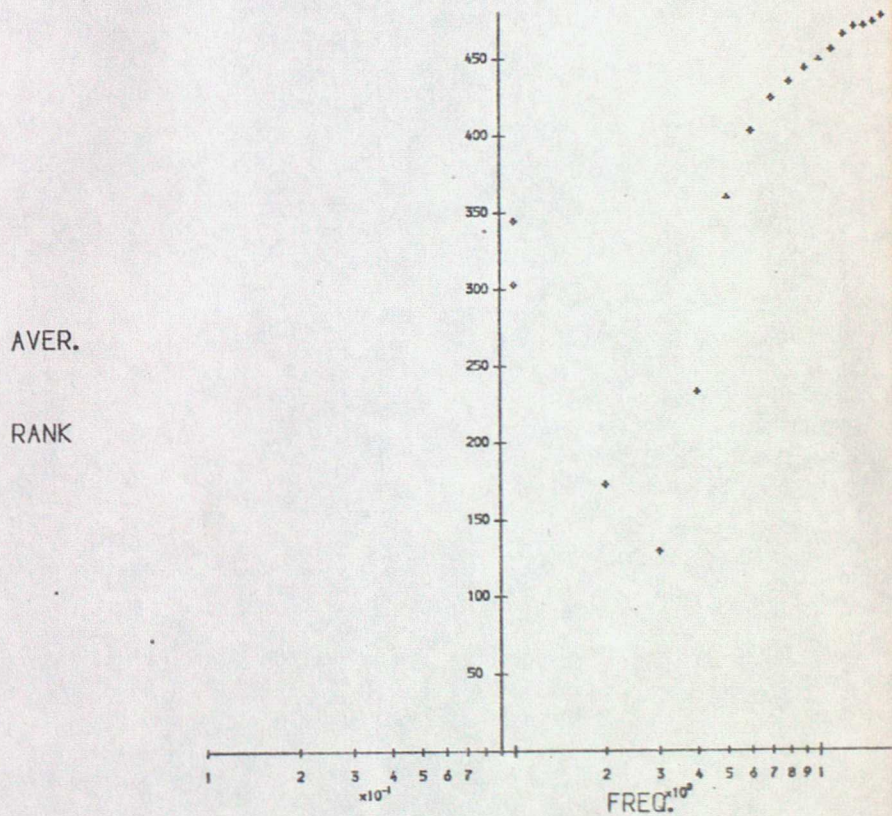
GRAPH II. 1 : (DVR) RANK VS FREQ. WITHIN REL. DOC. PLOT FOR THE
EVAN COLLECTION



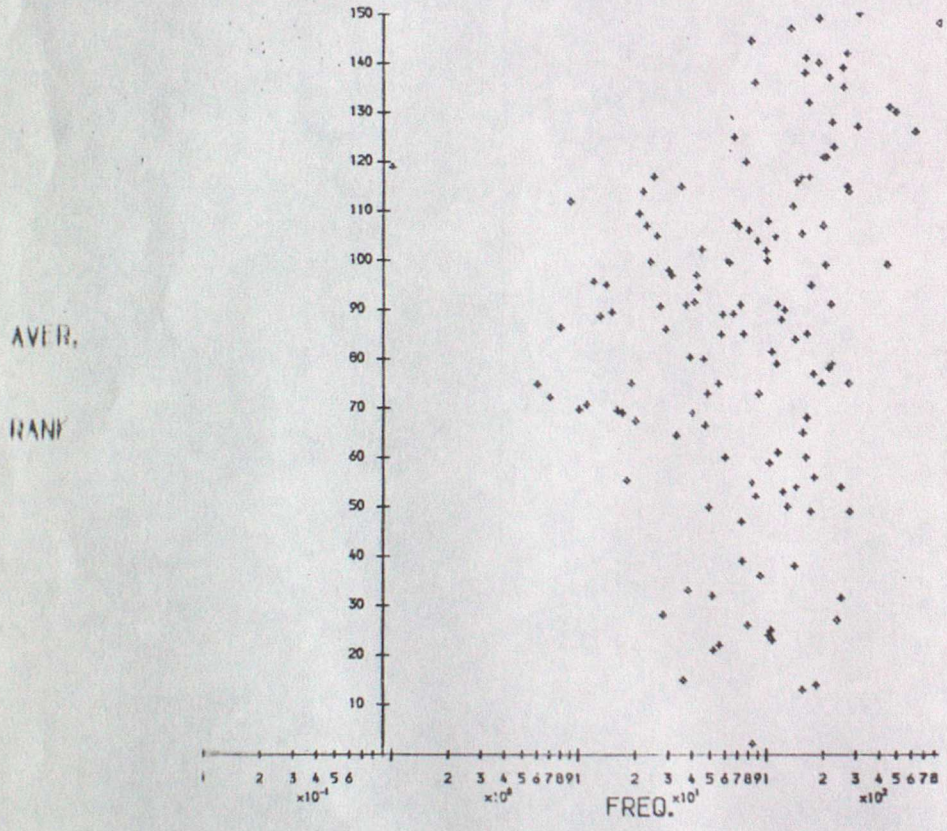
GRAPH II. 2 : (DVR) RANK VS FREQ. WITHIN REL. DOC. PLOT FOR
CRAN COLLECTION



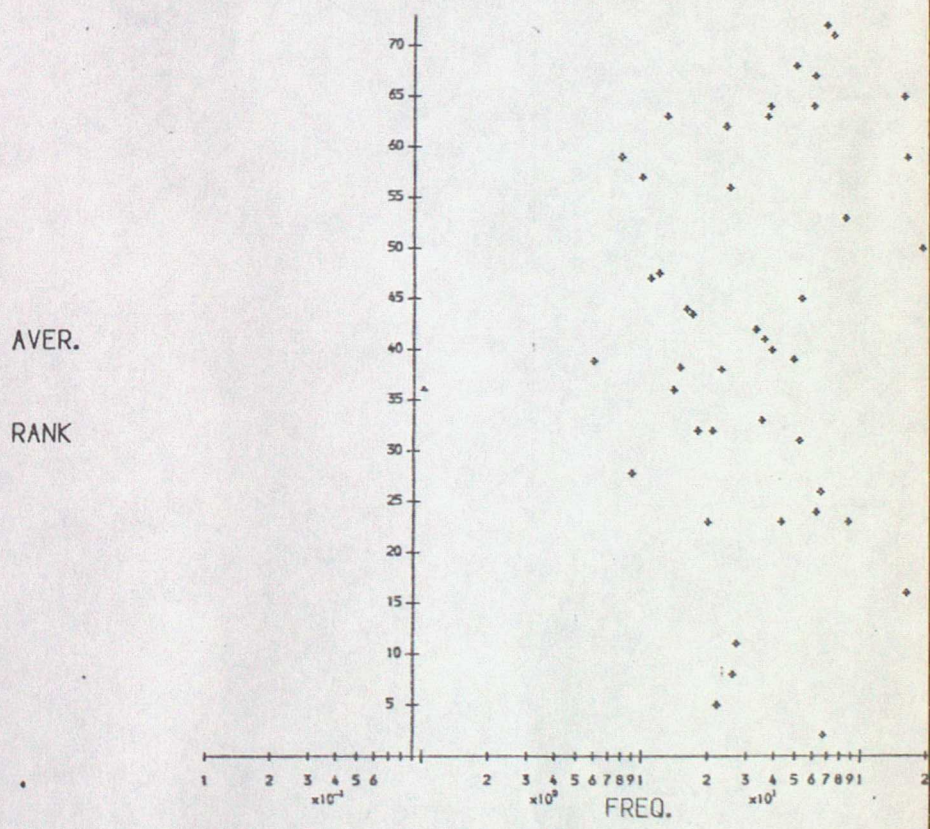
GRAPH II. 5 , (DVR) RANK VS FREQ. WITHIN REL. DOC. PLOT FOR THE LISA COLLECTION



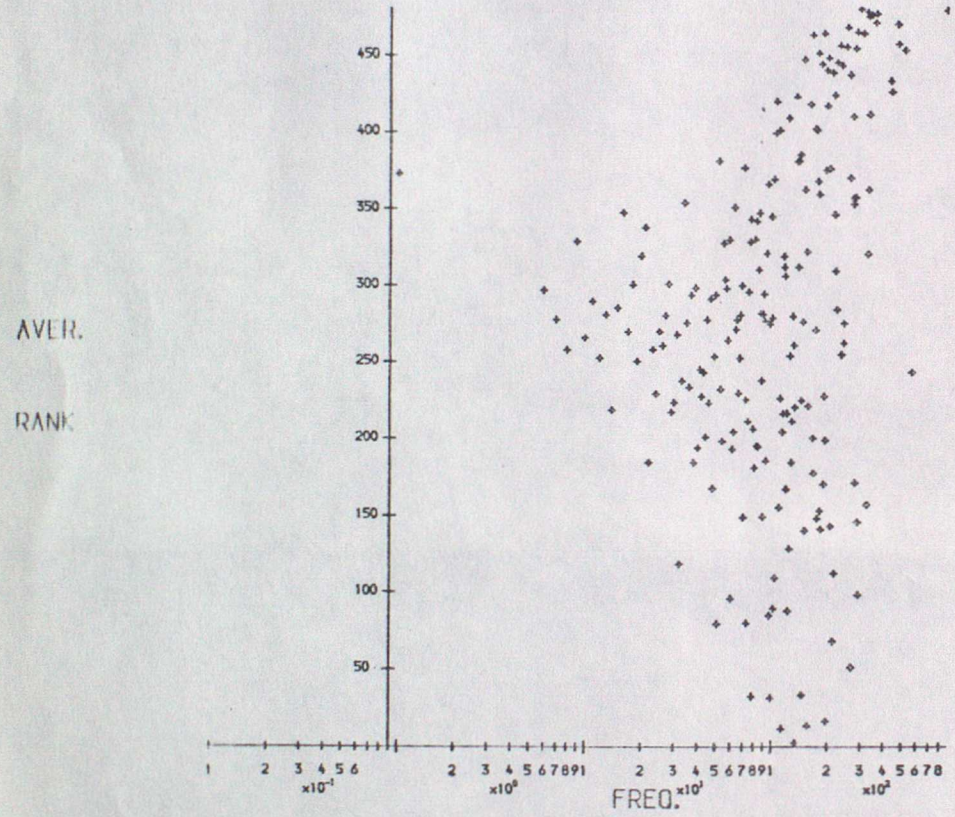
GRAPH II. 4 , (DVR) RANK VS FREQ. WITHIN REL. DOC. PLOT F HARD COLLECTION



GRAPH III2 : (DVR) RANK VS COLL. FREQ. PL FOR THE CRAN COLLECTION

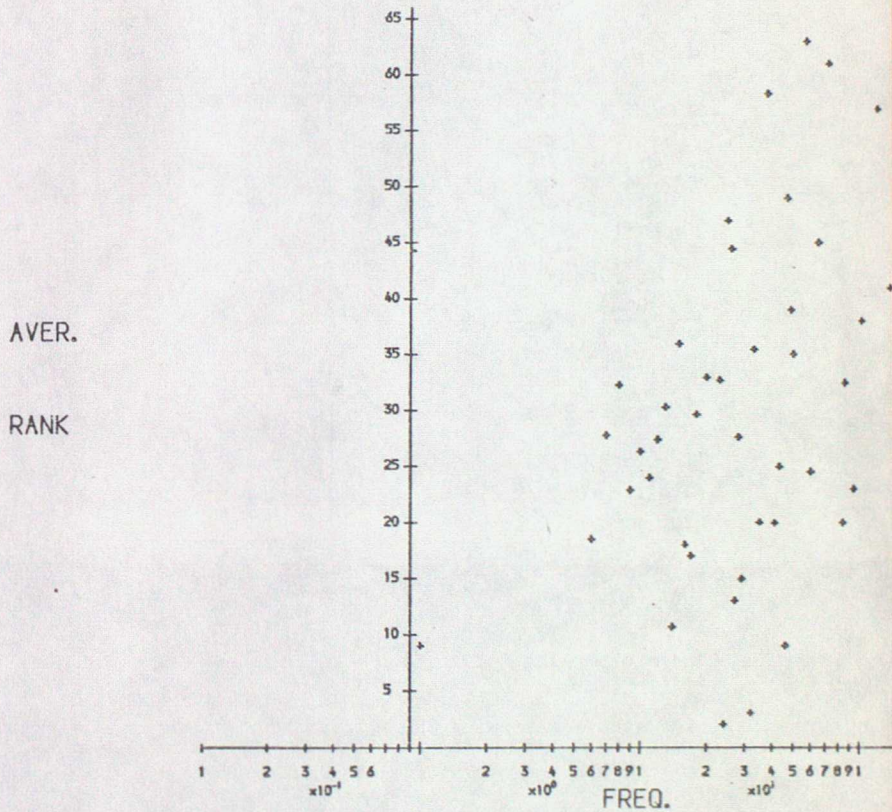


GRAPH III.1 : (DVR) RANK VS COLL. FREQ. PL FOR THE KEEN COLLECTION



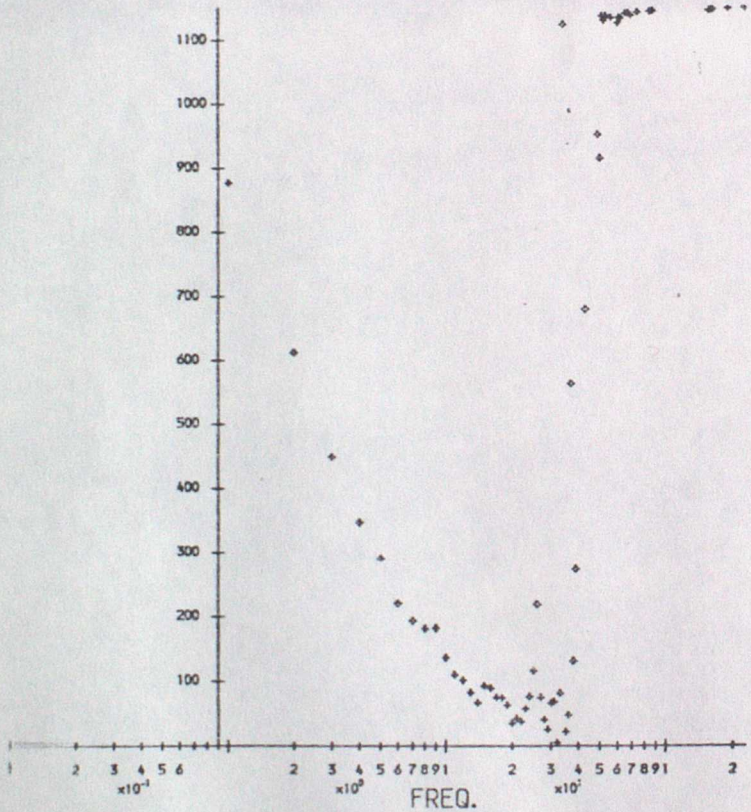
GRAPH IIIA : (DVR) RANK VS COLL. FREQ. PL FOR THE HARD COLLECTION

A-14

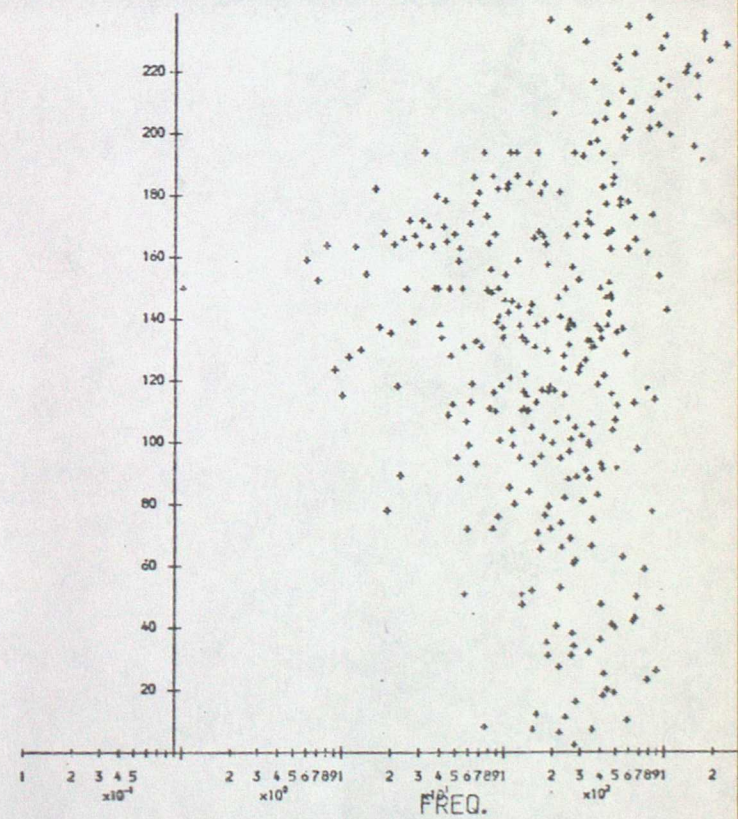


GRAPH IIIB : (DVR) RANK VS COLL. FREQ. PL FOR THE EVAN COLLECTION

A-13



GRAPH IV. 1 : AVER. (DVNRNR) RANK VS COLL. FREQ. PLOT FOR THE KEEN COLLECTION



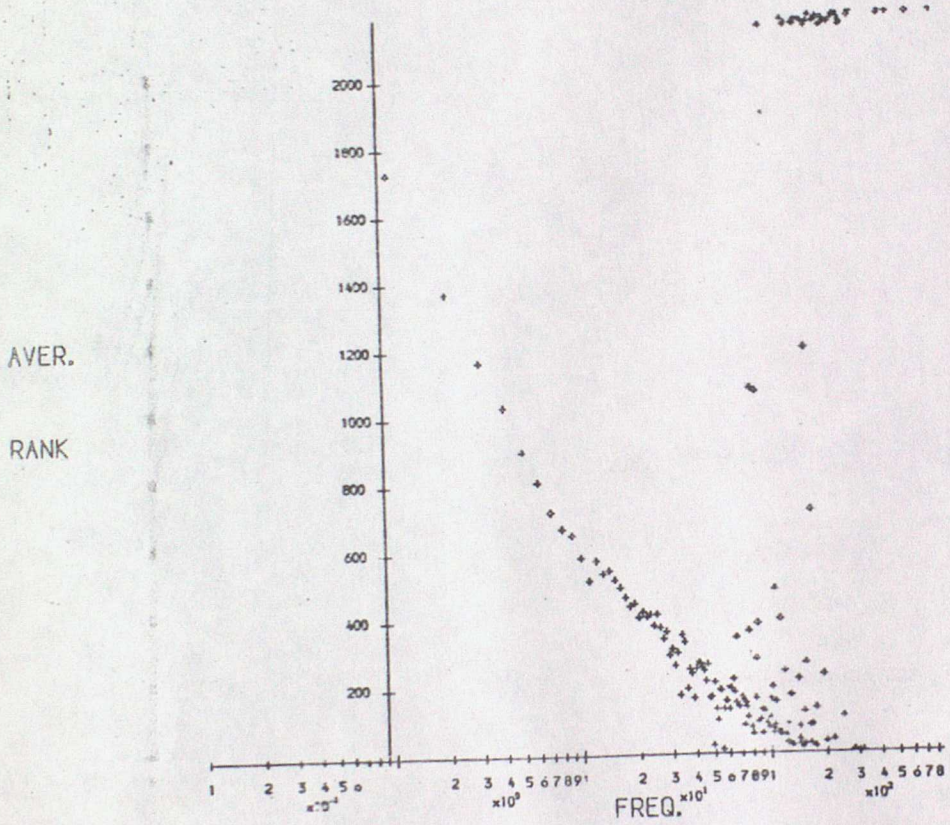
GRAPH IIIS : (DVR) RANK VS COLL. FREQ. PL FOR THE LISA COLLECTION

GRAPH IV. 2 : AVER. (DVNRNR) RANK VS COLL. FREQ. PLOT FOR THE CRAN COLLECTION

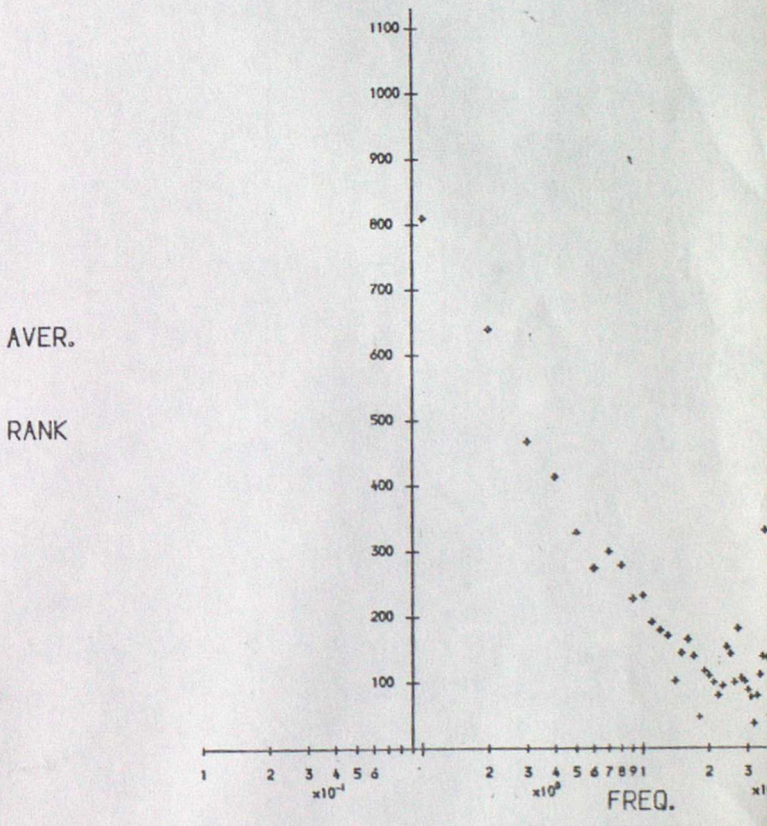
GRAPH IV. 3 : AVER. (DVNRNR) RANK VS COLL. FREQ. PLOT FOR THE EVAN COLLECTION

GRAPH IV. 4 : AVER. (DVNRNR) RANK VS COLL. FREQ. PLOT FOR THE HARD COLLECTION

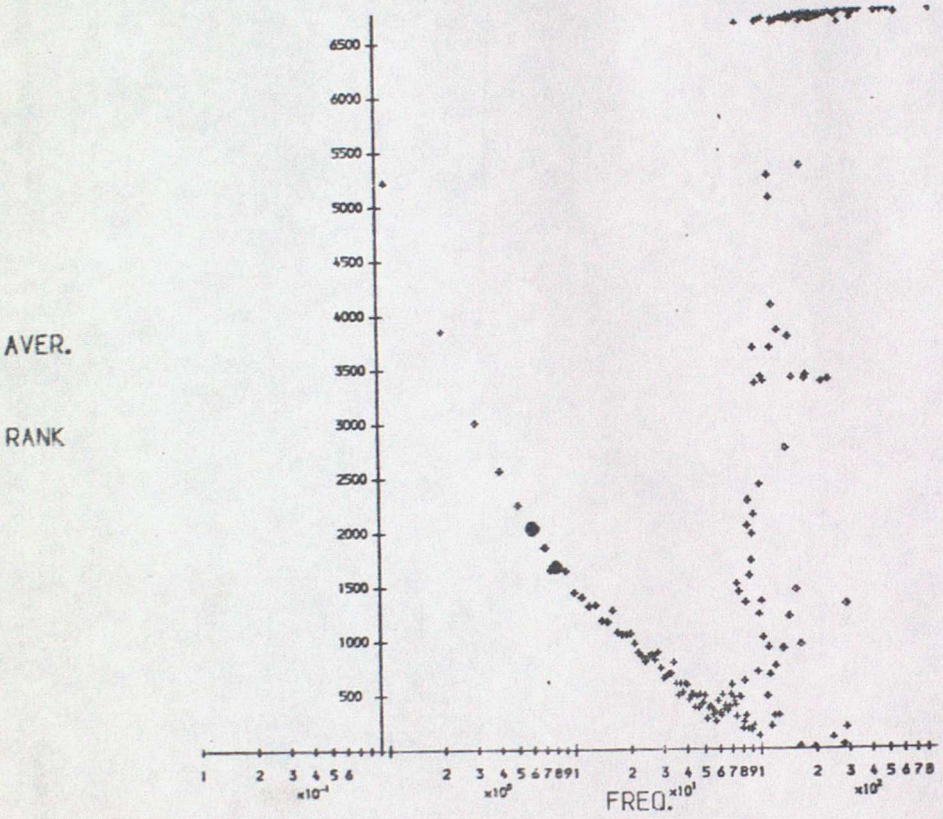
IV. 5 : AVER. (DVNRNR) RANK VS COLL. FREQ. PLOT FOR THE LISA COLLECTION



GRAPH V. 2 : AVER. (DVRNR) RANK VS COLL. FEQ. FOR THE CRAN COLLECTION FOR ALL TERMS

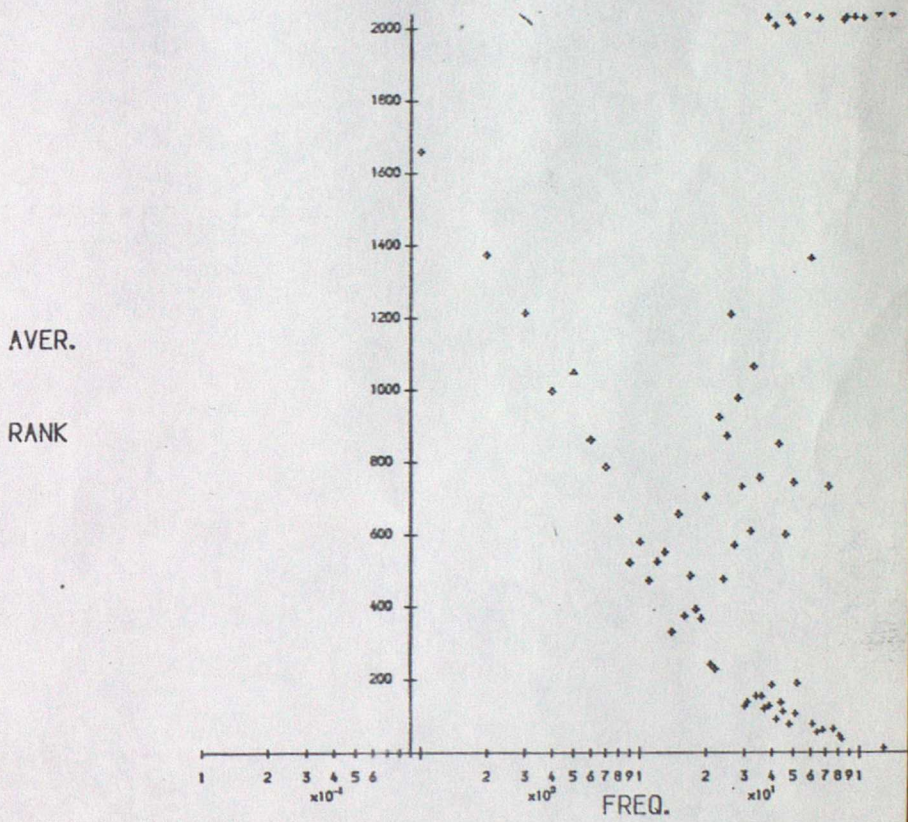


GRAPH V. 1 : AVER. (DVRNR) RANK VS COLL. FEQ. FO KEEN COLLECTION FOR ALL TERMS



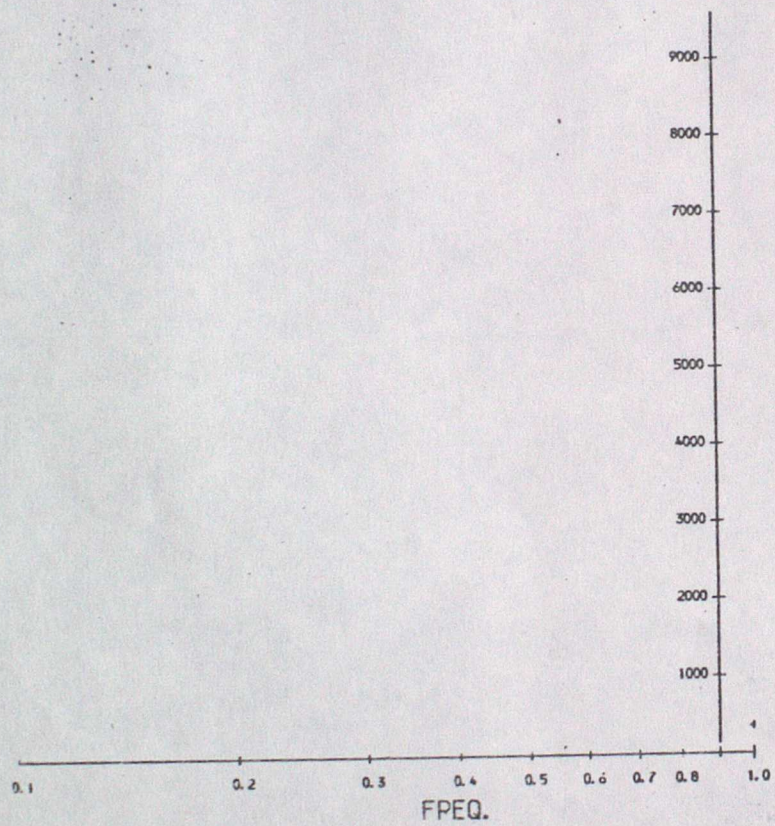
GRAPH V. 4 : AVER. (DVRNR) RANK VS COLL. FEQ. FOR THE HARD COLLECTION FOR ALL TERMS

A-24

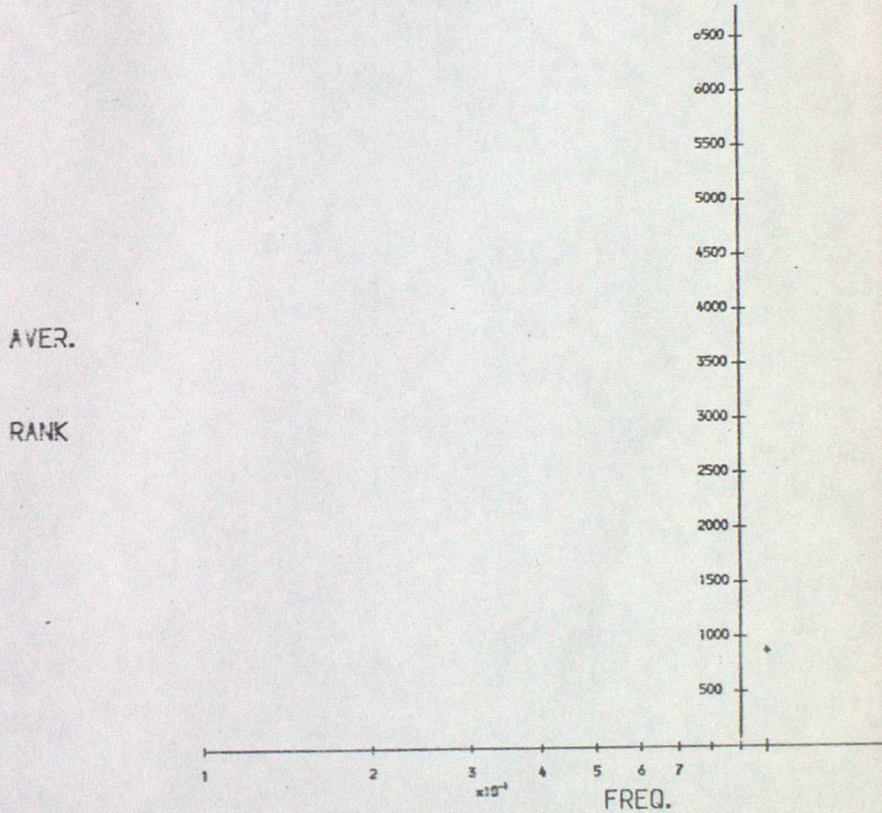


GRAPH V. 3 : AVER. (DVRNR) RANK VS COLL. FEQ. FOR THE EVAN COLLECTION FOR ALL TERMS

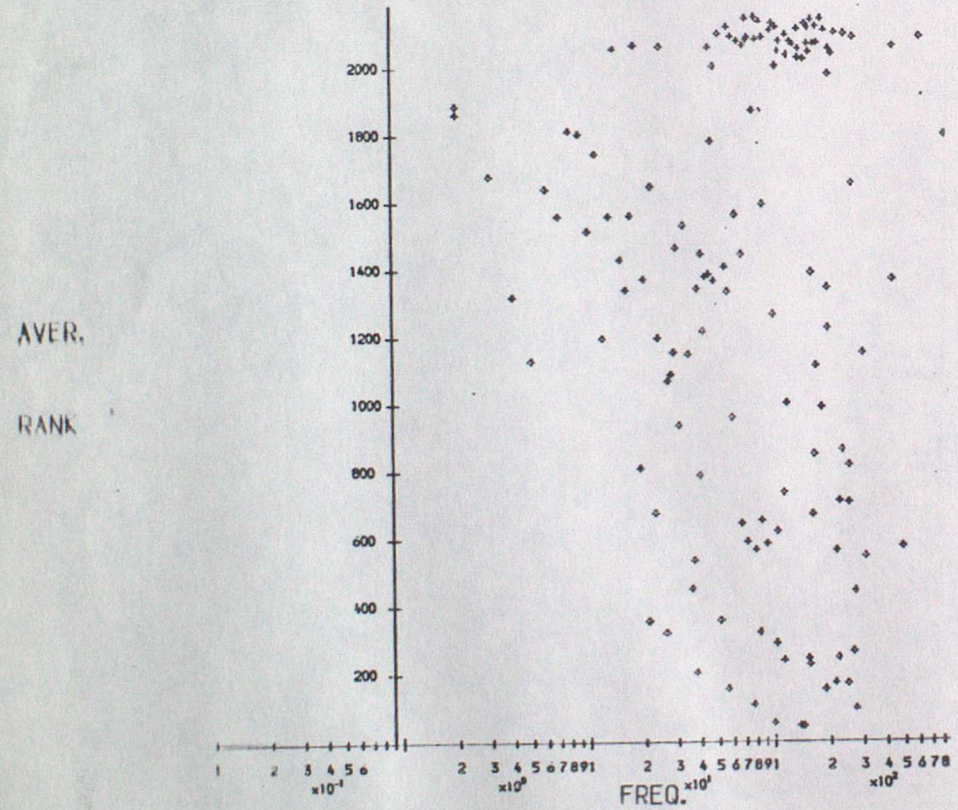
A-23



GRAPH VI. 5 : AVER. (DVRNR) RANK VS COLL. FEQ. FOR THE LISA COLLECTION FOR TERMS WHICH APPEAR ONLY IN REL.

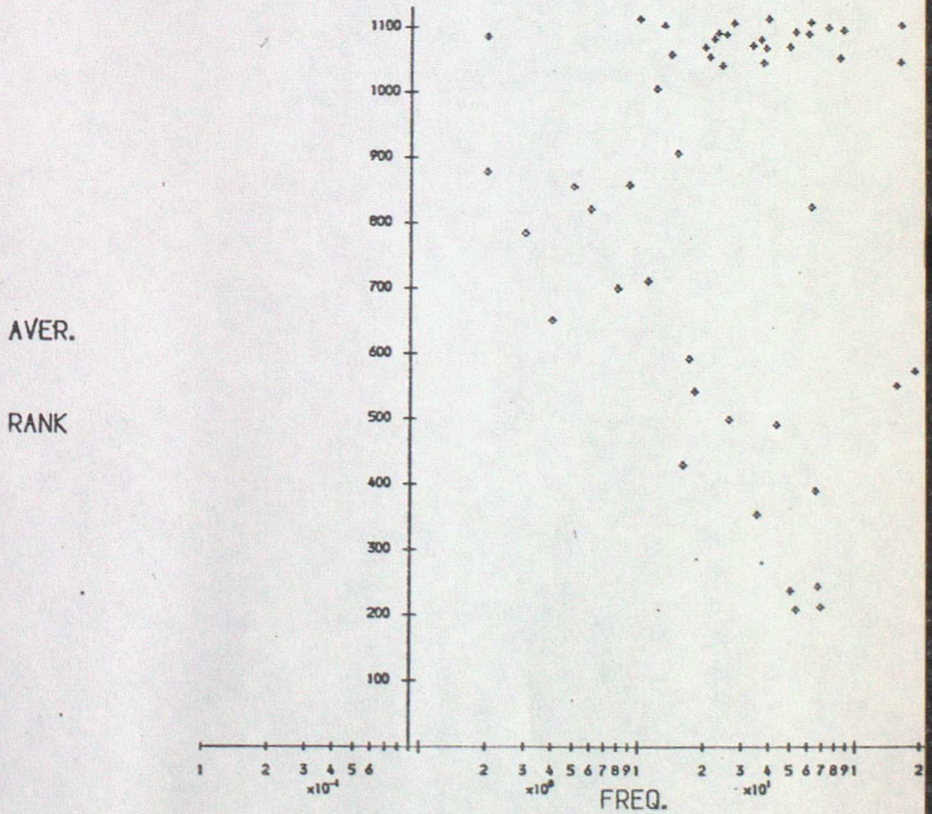


GRAPH VI. 4 : AVER. (DVRNR) RANK VS COLL. FEQ. FOR THE HARD COLLECTION FOR TERMS WHICH APPEAR ONLY IN REL.



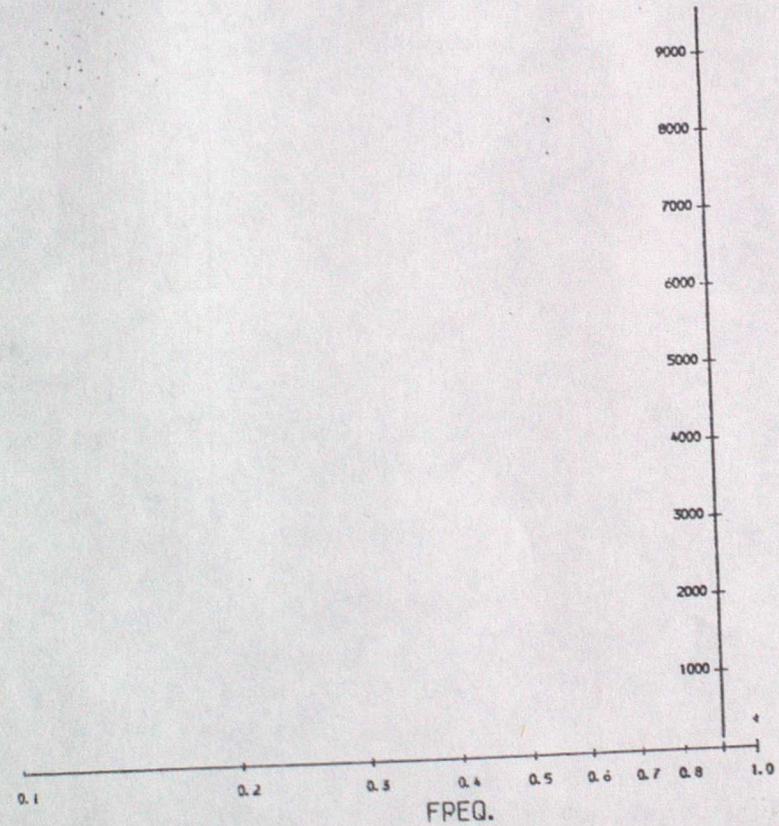
GRAPH VII.2 : AVER. (DVRNR) RANK VS COLL. EQ. FOR THE
CRAN COLLECTION
FOR TERMS WHICH APPEAR IN BOTH

A-32



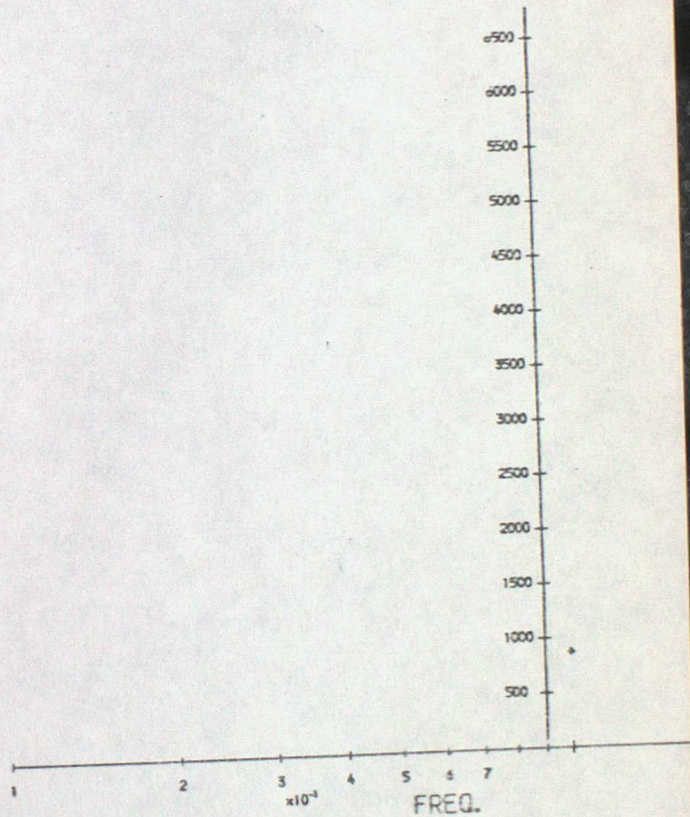
GRAPH VII.1 : AVER. (DVRNR) RANK VS COLL. EQ. FOR THE
KEEN COLLECTION
FOR TERMS WHICH APPEAR IN BOTH

A-31



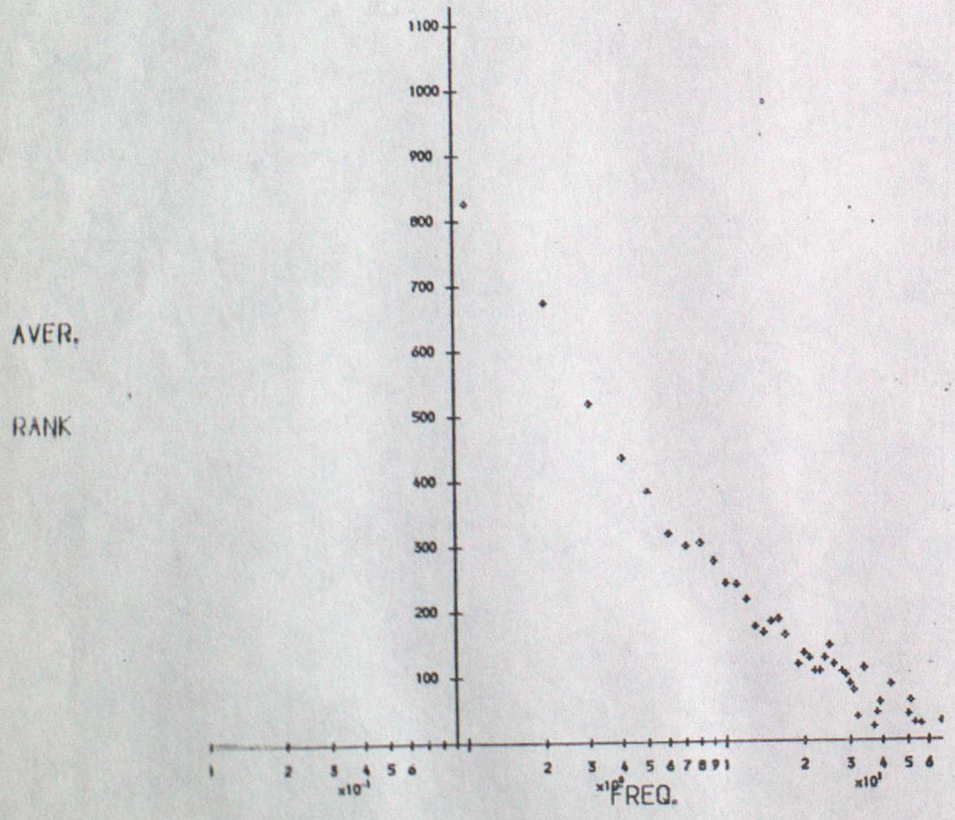
GRAPH VI. 5 : AVER. (DVRNR) RANK VS COLL. FEQ. FOR THE LISA COLLECTION FOR TERMS WHICH APPEAR ONLY IN REL.

A-30

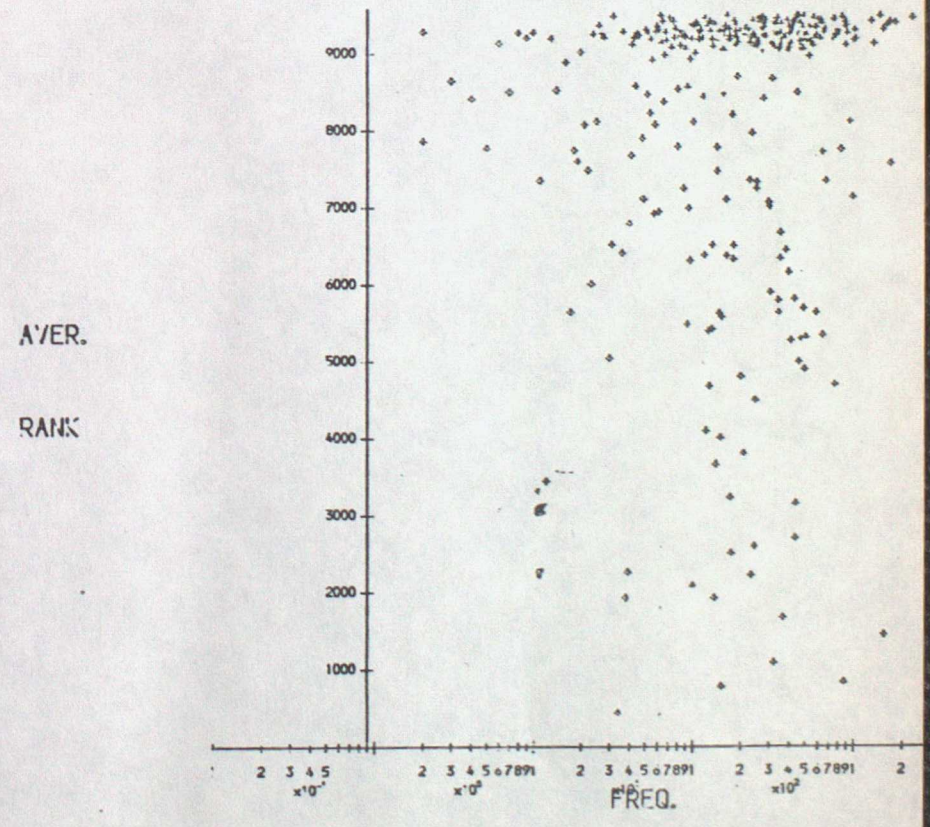


GRAPH VI. 4 : AVER. (DVRNR) RANK VS COLL. FEQ. FOR THE HARD COLLECTION FOR TERMS WHICH APPEAR ONLY IN REL.

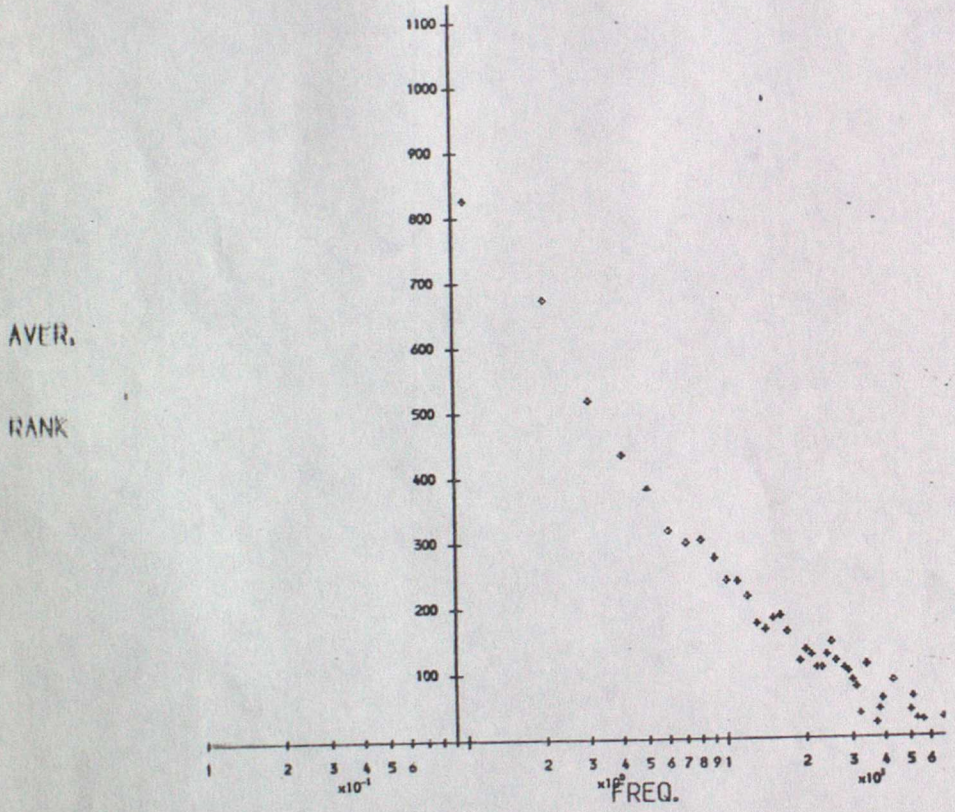
A-29



GRAPH VIII.1 : AVER. (DVRNR) RANK VS COLL. REQ. FOR THE KEEN COLLECTION
FOR TERMS WHICH APPEAR ONLY IN NON-REL.

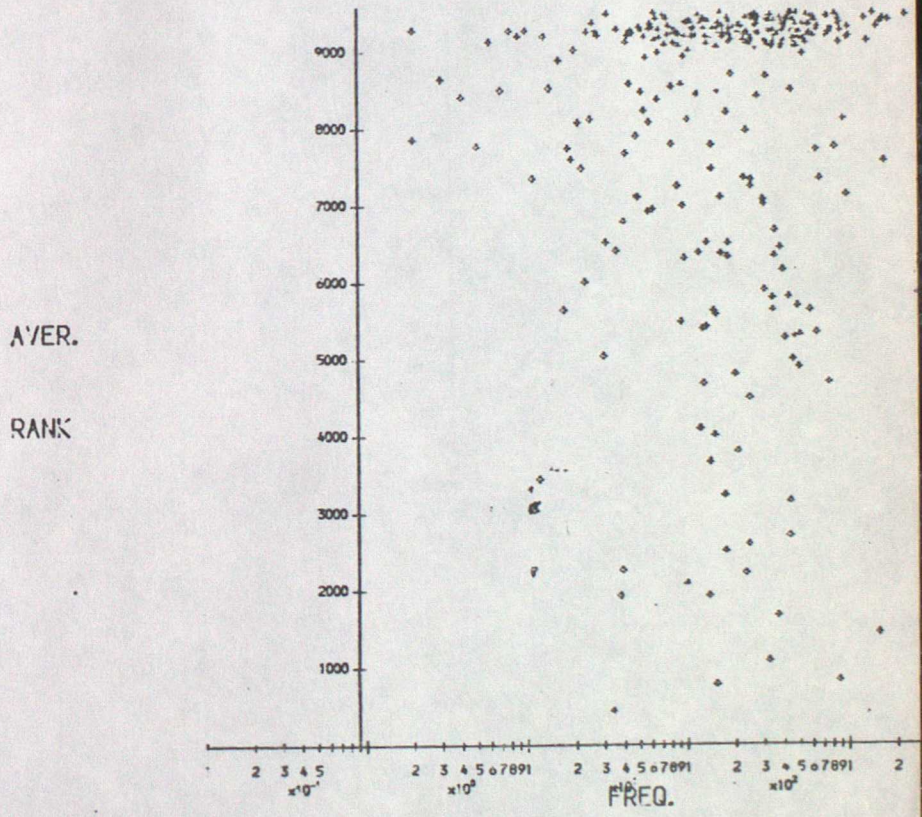


GRAPH VIII.5 : AVER. (DVRNR) RANK VS COLL. EQ. FOR THE LISA COLLECTION
FOR TERMS WHICH APPEAR IN BOTH



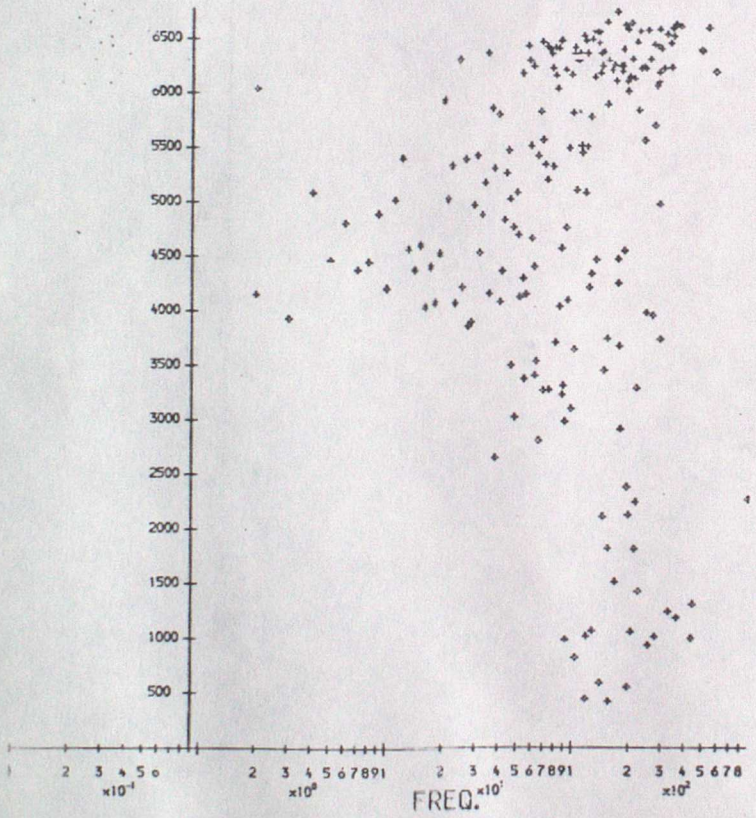
GRAPH VIII.1 : AVER. (DVRNR) RANK VS COLL. REQ. FOR THE KEEN COLLECTION FOR TERMS WHICH APPEAR ONLY IN NON-REL.

A-36



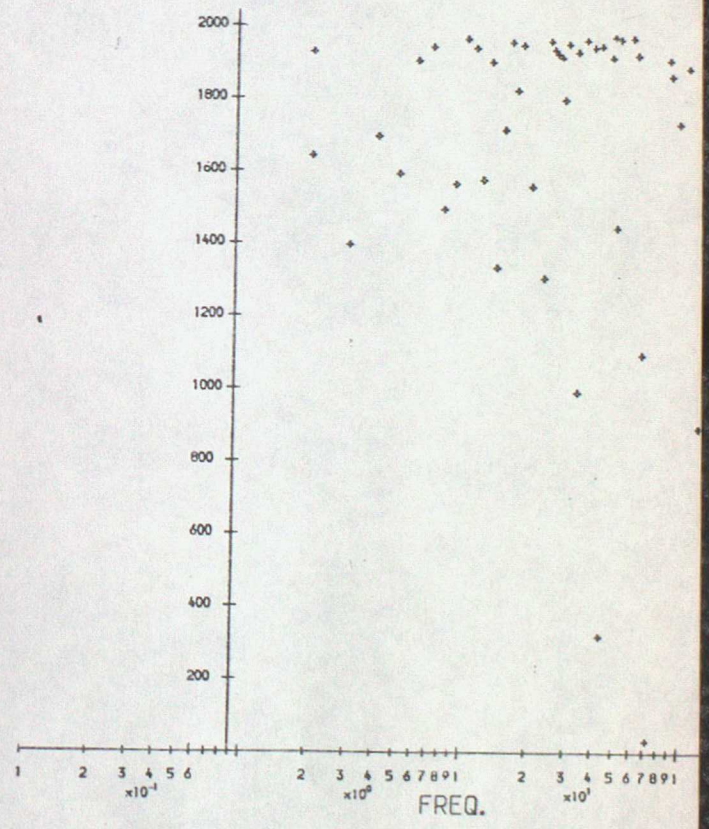
GRAPH VII5 : AVER. (DVRNR) RANK VS COLL. EQ. FOR THE LISA COLLECTION FOR TERMS WHICH APPEAR IN BOTH

A-35



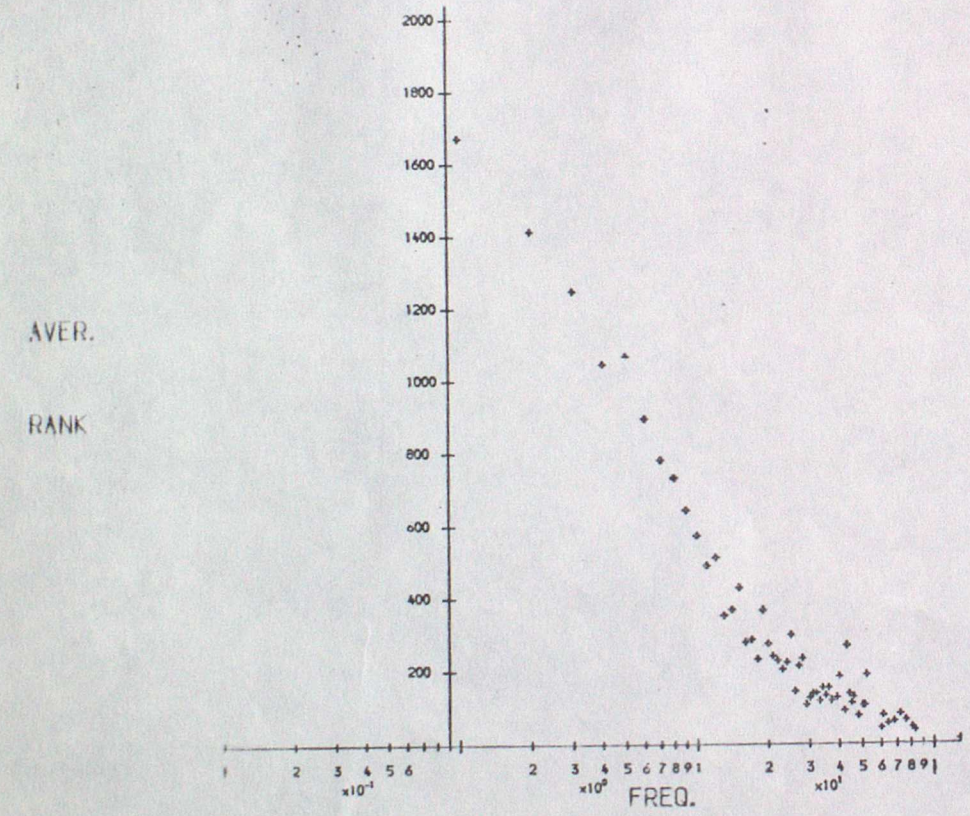
GRAPH VII2 : AVER. (DVRNR) RANK VS COLL. EQ. FOR THE HARD COLLECTION FOR TERMS WHICH APPEAR IN BOTH

A-34

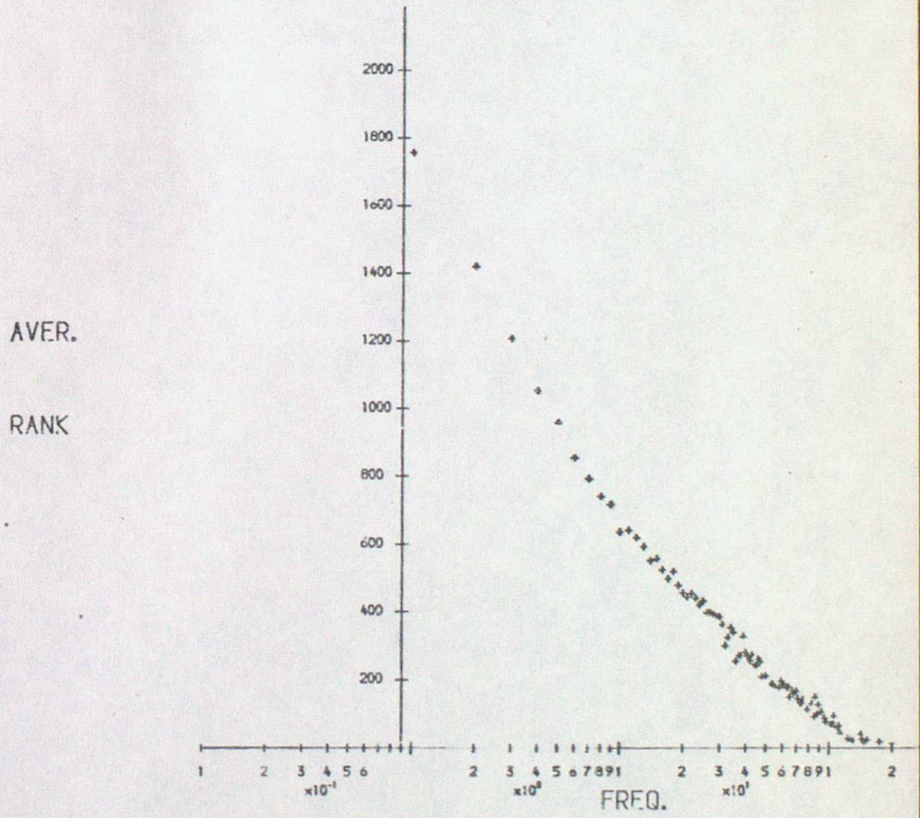


GRAPH VII3 : AVER. (DVRNR) RANK VS COLL. EQ. FOR THE EVAN COLLECTION FOR TERMS WHICH APPEAR IN BOTH

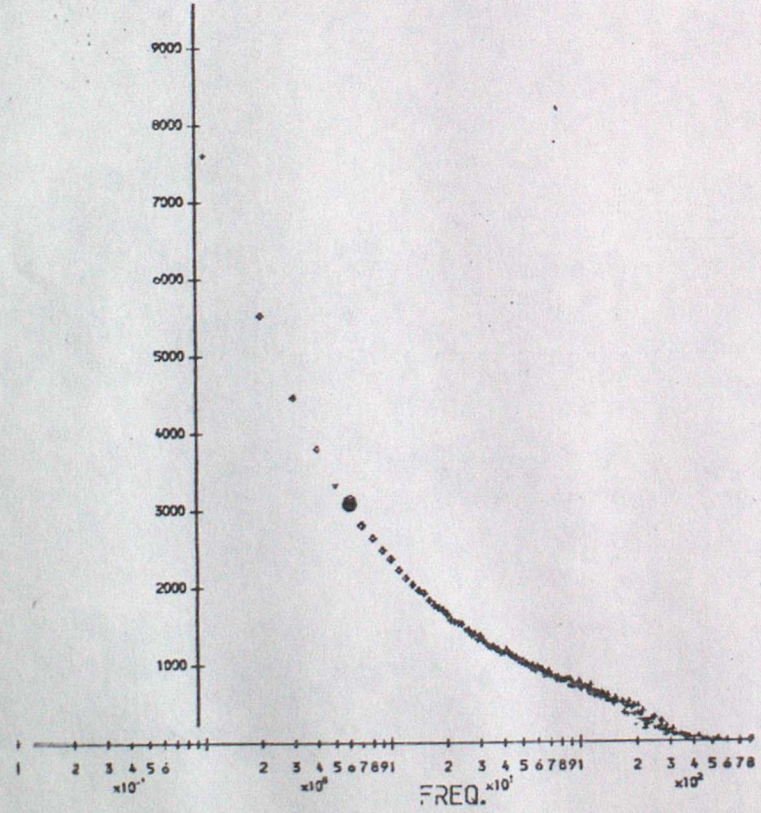
A-33



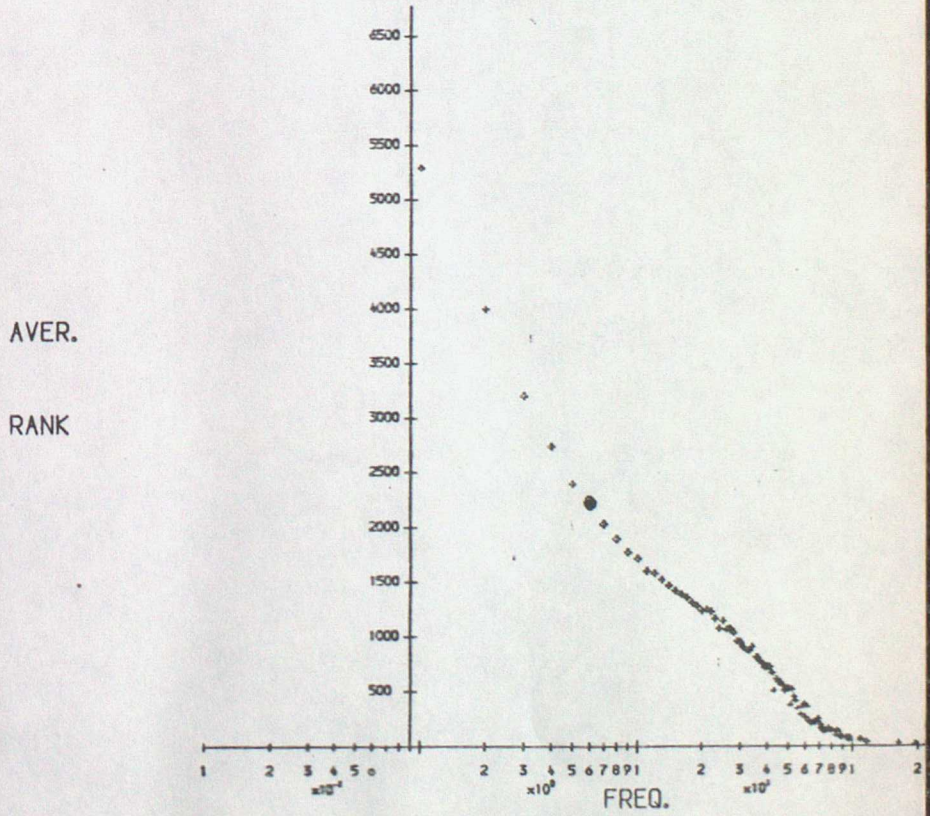
GRAPH VIII.1 : AVER. (DVRNR) RANK VS COLL. REQ. FOR THE EVAN COLLECTION FOR TERMS WHICH APPEAR ONLY IN NON-REL.



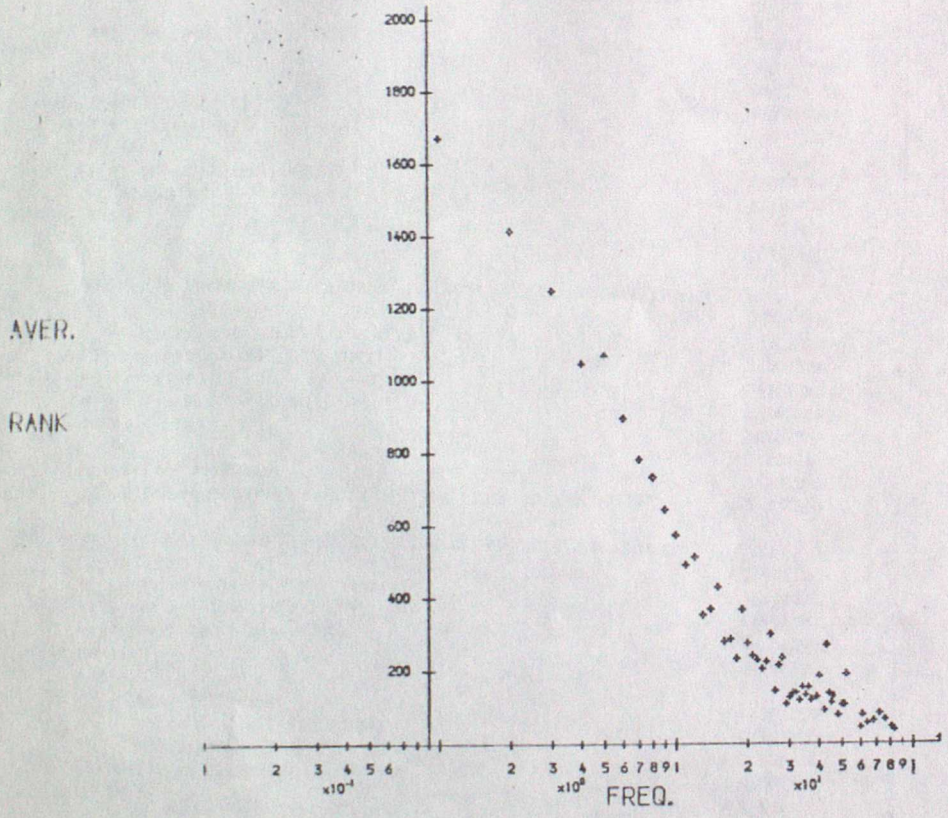
GRAPH VIII.2 : AVER. (DVRNR) RANK VS COLL. REQ. FOR THE CRAN COLLECTION FOR TERMS WHICH APPEAR ONLY IN NON-REL.



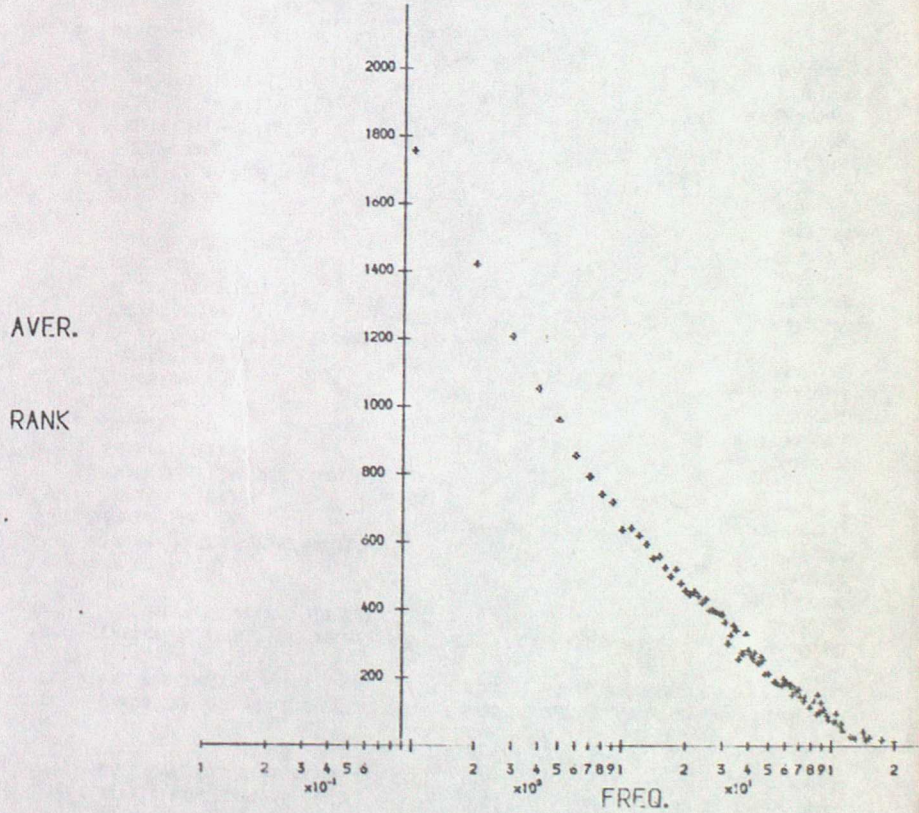
GRAPH V.11.5 : AVER. (DVRNR) RANK VS COLL. REQ. FOR THE LISA COLLECTION FOR TERMS WHICH APPEAR ONLY IN NON-REL.



GRAPH VIII.4 : AVER. (DVRNR) RANK VS COLL. REQ. FOR THE HARD COLLECTION FOR TERMS WHICH APPEAR ONLY IN NON-REL.



GRAPH VIII.1 : AVER. (DVRNR) RANK VS COLL. REQ. FOR THE EVAN COLLECTION FOR TERMS WHICH APPEAR ONLY IN NON-REL.



GRAPH VIII.2 : AVER. (DVRNR) RANK VS COLL. REQ. FOR THE CRAN COLLECTION FOR TERMS WHICH APPEAR ONLY IN NON-REL.

```
                                                                      DVA00770
      CALL DVSORT(M,DV,RNK,RMAX)                                      DVA00780
      DO 7 IJ=1,M                                                     DVA00790
    7 TF1(IJ)=NDC(IJ+1)-NDC(IJ)                                       DVA00800
      CALL DVPLOT(M,TF1,RNK,RMAX,1,FLN)


CC    READS THE RELEVANCE JUDGEMENT FOR A GIVEN QUERY, CALCULATES     DVA00820
CC    THE DVR VALUES, RANKS THEM, AND PLOTS THEM, AGAINST COLLECTION

CC    FREQUENCY WITHIN THE RELEVANT DOCUMENT SPACE, AND AGAINST       DVA00840
CC    THE OVER ALL COLLECTION FREQUENCY.                              DVA00850
                                                                      DVA00860
C     DO 60 JJ=1,5                                                    DVA00870
      PRINT*,'ENTER QUERY NUMBER'                                     DVA00880
      READ*,JJJ                                                       DVA00890
      DO 44 L=1,JJJ                                                   DVA00900
   44 READ(24,15)NNU,NNE,(RLD(K),K=1,NNE)                             DVA00910
   15 FORMAT(6I12)                                                    DVA00920
      DO 8 IJ=1,M                                                     DVA00930
    8 TF(IJ)=0                                                        DVA00940
      DO 25 I=1,NNE                                                   DVA00950
      DST(RLD(I))=1                                                   DVA00960
      DO 25 J=NTC(RLD(I)),NTC(RLD(I)+1)-1                             DVA00970
      TST(LT(J))=1                                                    DVA00980
      TF(LT(J))=TF(LT(J))+1                                           DVA00990
   25 CONTINUE                                                        DVA01000
      CALL DVCALC(M,IN,DVRR,1)                                        DVA01010
      JJJ=0                                                           DVA01020
      DO 27 I=1,M                                                     DVA01030
      IF(TF(I).NE.0)THEN                                              DVA01040
        JJJ=JJJ+1                                                     DVA01050
        TFF1(JJJ)=TF1(I)                                              DVA01060
        TFF(JJJ)=TF(I)                                                DVA01070
        DVRR(JJJ)=DVRR(I)                                             DVA01080
      ENDIF                                                           DVA01090
   27 CONTINUE


      CALL DVSORT(JJJ,DVRR,RNK,RMAX)                                  DVA01110
      CALL DVPLOT(JJJ,TFF,RNK,RMAX,2,FLN)                             DVA01120
      CALL DVPLOT(JJJ,TFF1,RNK,RMAX,3,FLN)                            DVA01130
                                                                      DVA01140
CC    PERFORMS THE ABOVE PROCEDURE FOR THE DVRR VALUES.               DVA01150
```



```
                                                                      DVA01160
      DO 45 I=1,IN                                                    DVA01170
      IF(DST(I) EQ 0)THEN                                             DVA01180
            DST(I)=2                                                  DVA01190
            DO 46 J=NTC(I),NTC(I+1)-1                                 DVA01200
   46       TST(LT(J))=2                                             DVA01210
      ENDIF                                                           DVA01220
   45 CONTINUE                                                        DVA01230
      CALL DVCALC(M,IN,DVNRNR,2)                                      DVA01240
      CALL DVSORT(M,DVNRNR,RNK,RMAX)                                  DVA01250
      CALL DVPLOT(M,TF1,RNK,RMAX,4,FLN)                               DVA01260
                                                                      DVA01270
CC    CALCULATE THE DVRNR VALUES USING THE VALUES OBTAINED FROM THE

CC    ABOVE TWO PROCEDURES, RANKS THEM, AND PLOTS THEM AGAINST        DVA01290
CC    COLLECTION FREQUENCY.                                           DVA01300
                                                                      DVA01310
      DO 45 I=1,M                                                     DVA01320
      A=DVR(I)-DVNRNR(I)                                              DVA01330
   45 DVNRNR(I)=A                                                     DVA01340
      CALL DVSORT(M,DVNRNR,RNK,RMAX)                                  DVA01350
      CALL DVPLOT(M,TF1,RNK,RMAX,5,FLN)                               DVA01360
                                                                      DVA01370
CC    PLOTS THE ABOVE GRAPH BUT ONLY FOR TERMS ONLY IN RELEVANT DOC.


      JJJ=0                                                           DVA01390
      DO 48 K=1,M                                                     DVA01400
      IF(TF(K) EQ TF1(K))THEN                                         DVA01410
      JJJ=JJJ+1                                                       DVA01420
      TFF(JJJ)=TF1(K)                                                 DVA01430
      RNKK(JJJ)=RNK(K)                                                DVA01440
      ENDIF                                                           DVA01450
   48 CONTINUE                                                        DVA01460
      CALL DVPLOT(JJJ,TFF,RNKK,RMAX,6,FLN)                            DVA01470
                                                                      DVA01480
CC    PLOTS THE RANK GRAPH BUT FOR TERMS WHICH ARE BOTH IN RELEVANT   DVA01490

CC    AND NON-RELEVANT

      JJJ=0                                                           DVA01510
      DO 49 K=1,M                                                     DVA01520
      IF(TF(K) LT TF1(K))THEN                                         DVA01530
         IF(TF(K) GT.0)THEN                                           DVA01540
                                                                      DVA01550
                                                                      DVA01560
```

```
CC    RANKING ROUTINE                                                  DVA02740
                                                                       DVA02750
      RNK(TRM(1))=1                                                    DVA02760
CC    WRITE(26,*)TRM(1),RNK(TRM(1))                                    DVA02770
      DO 29 I=2,M                                                      DVA02780
      IF(DV(I) EQ.DV(I-1))THEN                                         DVA02790
         RNK(TRM(I))=RNK(TRM(I-1))                                     DVA02800
      ELSE                                                             DVA02810
         RNK(TRM(I))=RNK(TRM(I-1))+1                                   DVA02820
      ENDIF                                                            DVA02830
      WRITE(26,*)TRM(I),RNK(TRM(I))                                    DVA02840
   29 CONTINUE                                                         DVA02850
      RMAX=RNK(TRM(M))                                                 DVA02860
      RETURN                                                           DVA02870
      END                                                              DVA02880
                                                                       DVA02890
CC    THIS SUBROUTINE CALCULATES THE AVERAGE RANK FOR TERMS OF THE
CC    RANK FREQUENCY; IT SORTS OUT DUPLICATES; AND IT PLOTS RANK       DVA02910
CC    AGAINST FREQUENCY. IT TAKES AS ITS INPUT THE TERM DISCRIMINATING
CC    VALUES AND TERM FREQUENCIES.                                     DVA02930
                                                                       DVA02940
      SUBROUTINE DVPLOT(M,T,RNK,RMAX,NUM,FLN)                          DVA02950
      INTEGER TF(20217),RNK(20217),TN(20217),T(20217)                  DVA02960
      LOGICAL ON(20217)                                                DVA02970
      CHARACTER*10,FLN                                                 DVA02980
      DIMENSION TFF(20217),RNKK(20217),RNNK(20217)                     DVA02990
      DATA ON/20217*.TRUE./                                            DVA03000
                                                                       DVA03010
CC    SORTH TERM FREQUENCY VALUES                                      DVA03020
                                                                       DVA03030
      DO 3 I=1,M                                                       DVA03040
      TF(I)=T(I)                                                       DVA03050
    2 TN(I)=I                                                          DVA03060
      DO 3 I=1,M-1                                                     DVA03070
      DO 3 J=I,M                                                       DVA03080
      IF(TF(I).LT.TF(J))THEN                                           DVA03090
         IT=TF(I)                                                      DVA03100
         TF(I)=TF(J)                                                   DVA03110
         TF(J)=IT                                                      DVA03120
         ITN=TN(I)                                                     DVA03130
         TN(I)=TN(J)                                                   DVA03140
         TN(J)=ITN                                                     DVA03150
```



```
FILE
                                                                       DVA02350
      SUBROUTINE STORE(NEC,LI,LY,JL,LK)                                DVA02360
      INTEGER NEC(LY+1),LI(JL),LK                                      DVA02370
      MC=1                                                             DVA02380
      DO 404 I=1,LY                                                    DVA02390
      READ(LK,1100)NU,NE,(LI(K),K=MC,MC+NE-1)                          DVA02400
      NEC(I)=MC                                                        DVA02410
      MC=MC+NE                                                         DVA02420
  404 CONTINUE                                                         DVA02430
      NEC(I)=MC                                                        DVA02440
 1100 FORMAT(6I12)                                                     DVA02450
      RETURN                                                           DVA02460
      END                                                              DVA02470
                                                                       DVA02480
CC    THIS SUBROUTINE SORTS THE DISCRIMINATING VALUES AND RANKS THEM.
CC    IT TAKES AS AN INPUT THE ACTUAL DISCRIMINATING VALUES (THE       DVA02500
CC    OUTPUT OF THE DVALUE SUBROUTINE)                                 DVA02510
                                                                       DVA02520
      SUBROUTINE DVSORT(M,D,RNK,RMAX)                                  DVA02530
      DIMENSION D(20217),DV(20217)                                     DVA02540
      INTEGER RNK(20217),TRM(20217)                                    DVA02550
                                                                       DVA02560
CC    SORTING ROUTINE (BUBBLE SORT)                                    DVA02570
                                                                       DVA02580
      DO 15 I=1,M                                                      DVA02590
      DV(I)=D(I)                                                       DVA02600
   15 TRM(I)=I
      :
      DO 20 I=1,M-1                                                    DVA02620
      DO 20 J=I+1,M                                                    DVA02630
      IF(DV(I).LT.DV(J))THEN                                           DVA02640
         DVST=DV(I)                                                    DVA02650
         DV(I)=DV(J)                                                   DVA02660
         DV(J)=DVST                                                    DVA02670
         ITRM=TRM(I)                                                   DVA02680
         TRM(I)=TRM(J)                                                 DVA02690
         TRM(J)=ITRM                                                   DVA02700
      ENDIF                                                            DVA02710
   20 CONTINUE                                                         DVA02720
                                                                       DVA02730
```

```
CC    MAIN PROGRAM; THIS PROGRM IS DESIGNED TO CALCULATE TERM          DVA00010
CC                  DISCRIMINATION VALUE FOR INDEXING TERMS USING      DVA00020
CC                  THE ALGORITHM PROPOSED BY ABDELMOULA EL-HAMDOUCHI
CC                  AND PETER WILLETT.                                 DVA00040
CC                  IT ALSO INCLUDES ADDITIONAL ROUTINES FOR: APPLYING
CC                  THE ALGORITHM ON DIFFERENT SETS OF REL.-REL.,      DVA00060
CC                  REL. - NON-REL., AND NON-REL. - NON-REL. DOCUMENT
CC                  DISTRIBUTION FOR A GIVEN QUERY;                    DVA00080
CC                  RANKING THE COMPUTED VALUES; COMPUTING AVERAGE
CC                  RANK FOR TERMS WITH THE SAME FREQUENCY; AND        DVA00100
CC                  PLOTING AVERAGE RANK AGAINST COLLECTION FREQUENCY.
CC                  THE PROGRAM CAN BE USED FOR DIFFERENT COLLECTIONS
CC                  BY SIMPLY ENTERING THE NAME OF THE COLLECTION,
AND
CC                  FOR ANY QUERY WITHIN THE COLLECTION BY SPECIFYING
CC                  THE QUERY NUMBER, UPON EXECUTION.                  DVA00150
                                                                       DVA00160
      DIMENSION DV(20217),DVRR(20217),DVNRNR(20217)                    DVA00170
      INTEGER RLD(800),DST,TST,ST                                      DVA00180
      COMMON /BLK1/ NTC(27362),LT(456000),H(54722),TO(20217),NDC(20218)

      COMMON /BLK2/ LD(456000),IH(20217),DST(27361),TST(20217)         DVA00200
      INTEGER TF(20217),TF1(20217),RNK(20217),RNKX(20217),TFF(20217)

      INTEGER TFF1(20217)                                              DVA00220
      CHARACTER*10,FLN                                                 DVA00230
                                                                       DVA00240
CC    THE FOLLOWING PROCEDURE SELECTS THE COLLECTION TO BE USED        DVA00250
                                                                       DVA00260
      PRINT *,'ENTER FILE NAME (USE UPPER CASE):  '                    DVA00270
      READ *, FLN                                                      DVA00280
      IF(FLN(1:2).EQ.'KE')THEN                                         DVA00290
            IN=800                                                     DVA00300
            M=1432                                                     DVA00310
            IL=7838                                                    DVA00320
            N=7838                                                     DVA00330
      ELSEIF(FLN(1:2).EQ.'EV')THEN                                     DVA00340
```



```
            IN=2542                                                    DVA00350
            M=3730                                                     DVA00360
            IL=16768                                                   DVA00370
            N=6300                                                     DVA00380
      ELSEIF(FLN(1:2).EQ.'HA')THEN                                     DVA00390
            IN=2472                                                    DVA00400
            M=8783                                                     DVA00410
            IL=89762                                                   DVA00420
            N=8783                                                     DVA00430
      ELSEIF(FLN(1:2).EQ.'CR')THEN                                     DVA00440
            IN=1400                                                    DVA00450
            M=2557                                                     DVA00460
            IL=40243                                                   DVA00470
            N=6000                                                     DVA00480
      ELSEIF(FLN(1:2).EQ.'LI')THEN                                     DVA00490
            IN=6004                                                    DVA00500
            M=13355                                                    DVA00510
            IL=238606                                                  DVA00520
            N=20000                                                    DVA00530
      ELSE                                                             DVA00540
         PRINT *,'ERROR IN FILE NAME'                                  DVA00550
         STOP                                                          DVA00560
      ENDIF                                                            DVA00570



CC    INITIALIZING THE PLOTTING DEVICE                                 DVA00590
                                                                       DVA00600
      CALL PAPER(1)                                                    DVA00610
                                                                       DVA00620
CC    CALLS THE ROUTINES USED TO READ DOCUMENT FILE AND TERM FILE      DVA00630
                                                                       DVA00640
      CALL STORE(NTC,LT,IN,IL,20)                                      DVA00650
      CALL STORE(NDC,LD,N,IL,21)                                       DVA00660
                                                                       DVA00670
CC    CALCULATES THE DV VALUES FOR THE ENTIRE COLLECTION, RANKS        DVA00680
CC    THESE VALUES AND PLOTS THEM AGAINST COLLECTION FREQUENCY.        DVA00690
                                                                       DVA00700
      DO 5 I=1,IN                                                      DVA00710
    5 DVI(I)=0                                                         DVA00720
      DO 6 I=1,M                                                       DVA00730
      TF1(I)=0                                                         DVA00740
    6 TDI(I)=0                                                         DVA00750
      CALL DVCALC(M,IN,DV,0)                                           DVA00760
```

```
      CALL MTIME(S)                                                     DVA01950
      DO 49 I=1,M                                                       DVA01960
      TO(I)=0                                                           DVA01970
      DVV(I)=0                                                          DVA01980
   20 H(I)=0                                                            DVA01990
      DO 10 I=1,M                                                       DVA02000
      IF((NDC(I+1)-NDC(I)).EQ.0)GO TO 10                                DVA02010
      IF(LST(I) .NE.ST)GO TO 10                                         DVA02020
    8 DO 10 J=NDC(I),NDC(I+1)-1                                         DVA02030
      IF(DST(LD(J)).EQ.ST)THEN                                          DVA02040
      W=NTC(LD(J)+1)-NTC(LD(J))                                         DVA02050
      TO(I)=TO(I)+1./SQRT(W)                                            DVA02060
      ENDIF                                                             DVA02070
   10 CONTINUE                                                          DVA02080
      DO 30 I=1,M                                                       DVA02090
      NJ=0                                                              DVA02100
      IF((NDC(I+1)-NDC(I)).EQ.0)GO TO 30                                DVA02110
      IF(LST(I) .NE.ST)GO TO 30                                         DVA02120
      DO 31 J=NDC(I),NDC(I+1)-1                                         DVA02130
      IF(DST(LD(J)).NE.ST)GO TO 31                                      DVA02140
      W=NTC(LD(J)+1)-NTC(LD(J))                                         DVA02150
      DO 31 K=NTC(LD(J)),NTC(LD(J)+1)-1                                 DVA02160
      IF(LT(K) )2,31,2                                                  DVA02170
    2 IF(H(LT(K)))4,4,5                                                 DVA02180

    4 NJ=NJ+1



      IH(NJ)=LT(K)                                                      DVA02200
    5 H(LT(K))=H(LT(K))-1./SQRT(W)+1./SQRT(W-1)                         DVA02210
   31 CONTINUE                                                          DVA02220
      DO 6 I=1,NJ                                                       DVA02230
      IHL=IH(I)                                                         DVA02240
      DVV(I)=DVV(I)+2*TO(IHL)*H(IHL)+(H(IHL))**2                        DVA02250
    6 H(IH(I))=0                                                        DVA02260
      DVV(I)=DVV(I)-(TO(I))**2                                          DVA02270
   30 CONTINUE                                                          DVA02280
      CALL MTIME(F)                                                     DVA02290
C     PRINT *,F-S                                                       DVA02300
      RETURN                                                            DVA02310
      END                                                               DVA02320
                                                                        DVA02330
CC    THIS SUBROUTINE READS IN BOTH THE DOCUMENT FILE AND THE TERM
```



```
        JJJ=JJJ+1                                                       DVA01570
        TFF(JJJ)=TF1(K)                                                 DVA01580
        RNKK(JJJ)=RNK(K)                                                DVA01590
      ENDIF                                                             DVA01600
    ENDIF                                                               DVA01610
 49 CONTINUE


    CALL DVPLOT(JJJ,TFF,RNK,RMAX,7,FLN)                                 DVA01630
                                                                        DVA01640
CC  PLOTS THE SAME GRAPH BUT FOR TERMS ONLY IN NON-RELEVANT DOC.

                                                                        DVA01660
    JJJ=0                                                               DVA01670
    DO 51 K=1,M                                                         DVA01680
    IF(TF(K).EQ.0)THEN                                                  DVA01690
    JJJ=JJJ+1                                                           DVA01700
    TFF(JJJ)=TF1(K)                                                     DVA01710
    RNKK(JJJ)=RNK(K)                                                    DVA01720
    ENDIF                                                               DVA01730
 51 CONTINUE                                                            DVA01740
    CALL DVPLOT(JJJ,TFF,RNKK,RMAX,8,FLN)                                DVA01750
    DO 50 I=1,M                                                         DVA01760
 50 TST(I)=0                                                            DVA01770
    DO 55 I=1,IN                                                        DVA01780
 55 DST(I)=0                                                            DVA01790
C 60 CONTINUE                                                           DVA01800
    CALL GREND                                                          DVA01810
    STOP                                                                DVA01820
    END                                                                 DVA01830
                                                                        DVA01840
CC  THIS SUBROUTINE CALCULATES THE DISCRIMINATING VALUES FOR EACH

CC  OF THE TERMS. IT TAKES AS ITS INPUT A SET OF DOCUMENTS AND A

CC  SET OF TERMS USED TO INDEX THESE DOCUMENTS.                         DVA01870
                                                                        DVA01880
    SUBROUTINE DVCALC(M,IN,DVV,ST)                                      DVA01890
    INTEGER DST,TST,ST                                                  DVA01900
    COMMON /BLK1/ NTC(27362),LT(456000),H(54722),TO(20217),NDC(20218)

    COMMON /BLK2/ LD(456000),IH(20217),DST(27361),TST(20217)            DVA01920
    DIMENSION DVV(20217)                                                DVA01930
    REAL*8 S,F                                                          DVA01940
```

```
      1L. DOC. PLOT FOR THE ')                                          DVA03960
       ELSEIF(NUM.EQ.3)THEN                                             DVA03970
         CALL PLOTCS(0.05,0.1,'GRAPH III.  : (DVR) RANK VS COLL. FREQ.
     PL.
      1 FOR THE ')                                                      DVA03990
       ELSEIF(NUM.EQ.4)THEN                                             DVA04000
         CALL PLOTCS(0.05,0.1,'GRAPH IV.  : AVER. (DVNRNR) RANK VS COLL.

      1FREQ. PLOT FOR THE ')                                            DVA04020
       ELSEIF(NUM.EQ.5)THEN                                             DVA04030
         CALL PLOTCS(0.05,0.1,'GRAPH V.   : AVER. (DVRNR) RANK VS COLL.

      1EQ. FOR THE ')                                                   DVA04050
         CALL PLOTCS(0.25,0.01,'FOR ALL TERMS')                         DVA04060
       ELSEIF(NUM.EQ.6)THEN                                             DVA04070
         CALL PLOTCS(0.05,0.1,'GRAPH VI.  : AVER. (DVRNR) RANK VS COLL.

      1EQ. FOR THE ')                                                   DVA04090
         CALL PLOTCS(0.25,0.01,'FOR TERMS WHICH APPEAR ONLY IN REL.')

       ELSEIF(NUM.EQ.7)THEN                                             DVA04110
         CALL PLOTCS(0.05,0.1,'GRAPH VII.  : AVER. (DVRNR) RANK VS COLL.

      1EQ. FOR THE ')                                                   DVA04130
         CALL PLOTCS(0.25,0.01,'FOR TERMS WHICH APPEAR IN BOTH')        DVA04140
       ELSE                                                             DVA04150
         CALL PLOTCS(0.03,0.1,'GRAPH VIII.  : AVER. (DVRNR) RANK VS
     COLL.I
      1REQ. FOR THE ')                                                  DVA04170
         CALL PLOTCS(0.25,0.01,'FOR TERMS WHICH APPEAR ONLY IN NON-REL.')DVA04180
       ENDIF                                                            DVA04190
       CALL PLOTCS(0.25,0.05,FLN)                                       DVA04200
       IF(FLN(1:2).EQ.'KE')THEN                                         DVA04210
            CALL PLOTCS(0.20,0.1,'1')                                   DVA04220
       ELSEIF(FLN(1:2).EQ.'EV')THEN                                     DVA04230
            CALL PLOTCS(0.20,0.1,'3')                                   DVA04240
       ELSEIF(FLN(1:2).EQ.'CR')THEN                                     DVA04250
            CALL PLOTCS(0.20,0.1,'2')                                   DVA04260
       ELSEIF(FLN(1:2).EQ.'HA')THEN                                     DVA04270
            CALL PLOTCS(0.20,0.1,'4')                                   DVA04280
       ELSE                                                             DVA04290
            CALL PLOTCS(0.20,0.1,'5')                                   DVA04300
       ENDIF                                                            DVA04310
       CALL PLOTCS(0.5,0.05,'COLLECTION')                               DVA04320
```



```
      CALL GTRMAG(10)                                                   DVA04330
      CALL PSPACE(0.2,0.95,0.2,0.95)                                    DVA04340
      CALL ABORIG(0.9,0.0)                                              DVA04350
      CALL MAPXL(0.1,FMAX,0.1,RMAX)                                     DVA04360
      CALL AXEXL                                                        DVA04370
      CALL PYPLOT(TFF,RNNK,1,JJ,0)                                      DVA04380
      CALL MAP(0.,1.,0.,1.)                                             DVA04390
      CALL FRAME                                                        DVA04400
      ENDIF                                                             DVA04410
      RETURN                                                            DVA04420
      END
```

```
      ENDIF                                                          DVA03160
    3 CONTINUE                                                       DVA03170
                                                                     DVA03180
CC    CALCULATES AVERAGE RANK                                        DVA03190
                                                                     DVA03200
      FMAX=TF(1)                                                     DVA03210
      DO 5 I=1,M                                                     DVA03220
    5 RNKK(I)=RNK(I)                                                 DVA03230
      I=1                                                            DVA03240
   10 SUM=RNKK(TN(I))                                                DVA03250
      NO=1                                                           DVA03260
      J=I+NO                                                         DVA03270
   16 IF(TF(I).EQ.TF(J))THEN                                         DVA03280

      SUM=SUM+RNKK(TN(J))                                            DVA03290
      NO=NO+1

      IF(J.EQ.M)GO TO 17                                             DVA03310
      J=I+NO                                                         DVA03320
      GO TO 16                                                       DVA03330
      ENDIF                                                          DVA03340
   17 DO 18 K=I,J-1                                                  DVA03350
   18 RNKK(TN(K))=SUM/NO                                             DVA03360
      I=J                                                            DVA03370
      IF(I.LT.M)GO TO 10                                             DVA03380
      IF(TF(M-1).NE.TF(M))RNKK(TN(M))=RNKK(TN(M))                    DVA03390
                                                                     DVA03400
CC    DELETES ZERO VALUES (IF ANY)                                   DVA03410
                                                                     DVA03420
      K=0                                                            DVA03430
      DO 20 I=1,M                                                    DVA03440
      IF(TF(I).GT.0)THEN                                             DVA03450
        IF(RNKK(I).GT.0.0)THEN                                       DVA03460
          K=K+1                                                      DVA03470
          TF(K)=TF(I)                                                DVA03480
          RNKK(K)=RNKK(I)                                            DVA03490
        ENDIF                                                        DVA03500
      ENDIF                                                          DVA03510
   20 CONTINUE                                                       DVA03520
                                                                     DVA03530
CC    DELETES DUPLICATES                                             DVA03540
                                                                     DVA03550
      FGAP=(FMAX/110.0)*(10**(-6))                                   DVA03560
```



```
      RGAP=(RMAX/110.0)*0.00002                                      DVA03570
      DO 22 I=1,K-1                                                  DVA03580
        IF(ON(I))THEN                                                DVA03590
        DO 21 J=I+1,K                                                DVA03600
          IF(ON(J))THEN                                              DVA03610
           ON(J)=(ABS(ALOG10(FLOAT(TF(I)))-ALOG10(FLOAT(TF(J)))).GT.

     1          RGAP).OR.                                           DVA03630
     2       (ABS(ALOG10(RNKK(TN(I)))-ALOG10(RNKK(TN(J))))          DVA03640
     3         GT.RGAP)                                            DVA03650
          ENDIF                                                      DVA03660
   21   CONTINUE                                                    DVA03670
        ENDIF                                                        DVA03680
   22 CONTINUE                                                       DVA03690
      DO 24 I=1,K                                                    DVA03700
      IF(IF(I).GT.5)THEN                                             DVA03710
        ON(I)=.TRUE.                                                 DVA03720
      ENDIF                                                          DVA03730
   23 CONTINUE                                                       DVA03740
      JJ=0                                                           DVA03750
      DO 24 I=1,K                                                    DVA03760
      IF (ON(I))THEN                                                 DVA03770
          JJ=JJ+1                                                    DVA03780
          TF(JJ)=TF(I)                                               DVA03790
          RNKK(JJ)=RNKK(TN(I))                                       DVA03800
      ENDIF                                                          DVA03810
   24 CONTINUE



                                                                     DVA03830
CC    PLOTS AVERAGE RANK AGAINST COLLECTION FREQUENCY                 DVA03840
      IF(JJ.GT.0)THEN                                                DVA03850
      CALL PAPACE(0.0,0.95,0.0,0.95)                                 DVA03860
      CALL GIRMAG(18)                                                DVA03870
      CALL PLOTCS(0.01,0.65,'AVER.')                                 DVA03880
      CALL PLOTCS(0.01,0.55,'RANK')                                  DVA03890
      CALL PLOTCS(0.65,0.15,'FREQ.')                                 DVA03900
      IF(NUM.EQ.1)THEN                                               DVA03910
        CALL PLOTCS(0.05,0.1,'GRAPH I.  : AVER. (DV) RANK VS COLL.
     FREQ.
     1   PLOT FOR THE ')                                             DVA03930
      ELSEIF(NUM.EQ.2)THEN                                           DVA03940
        CALL PLOTCS(0.05,0.1,'GRAPH II.  : (DVR) RANK VS FREQ. WITHIN
     RE.
```